

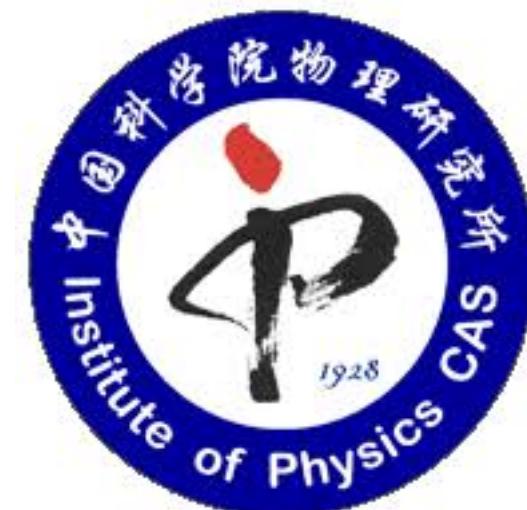
# Tensor Networks for Generative Modeling

**From Boltzmann machines to Born machines, and back**

Lei Wang (王磊)

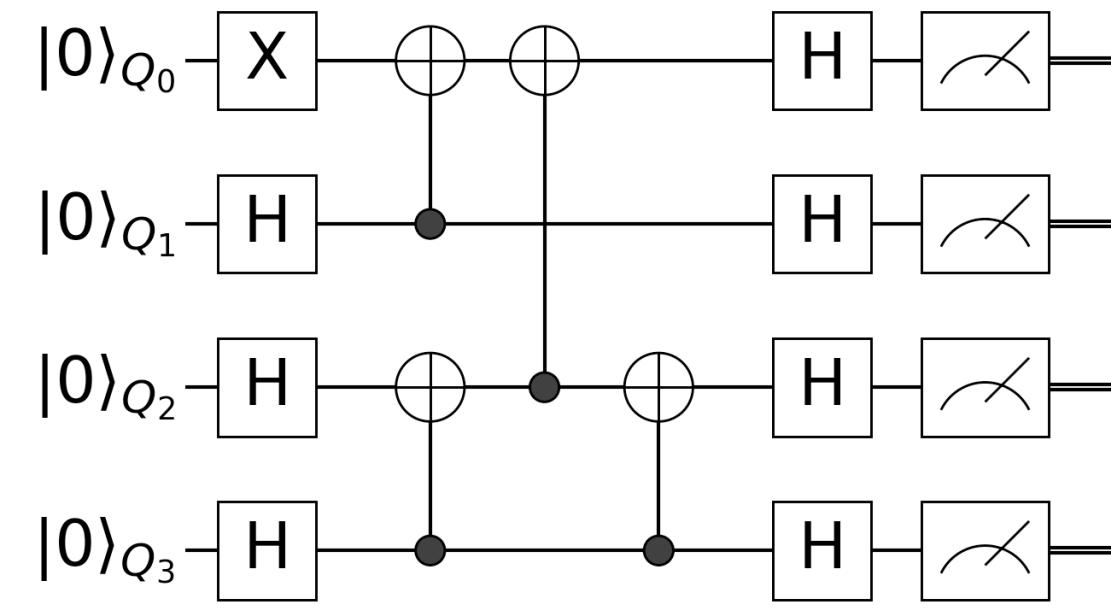
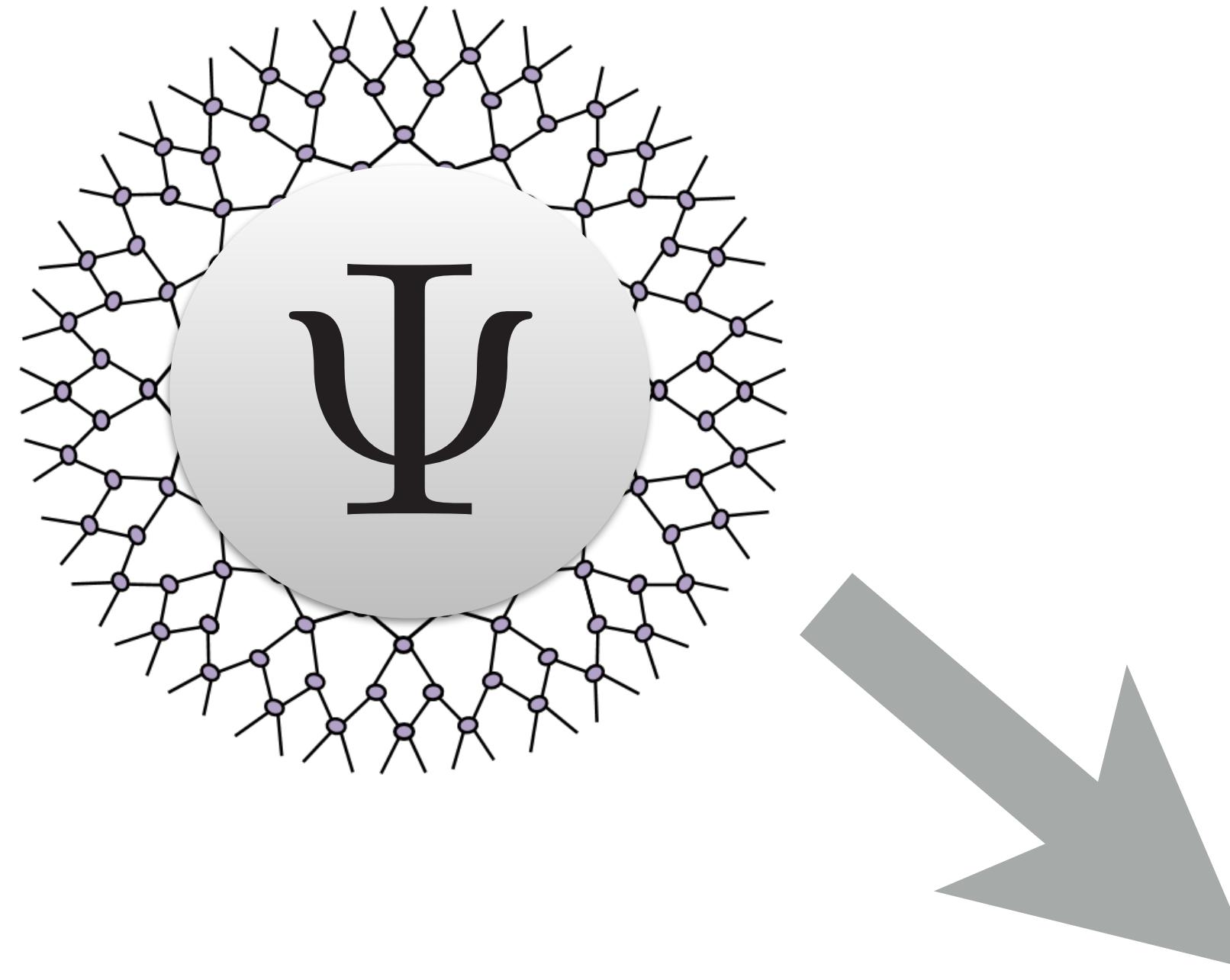
<https://wangleiphy.github.io>

Institute of Physics, Beijing  
Chinese Academy of Sciences



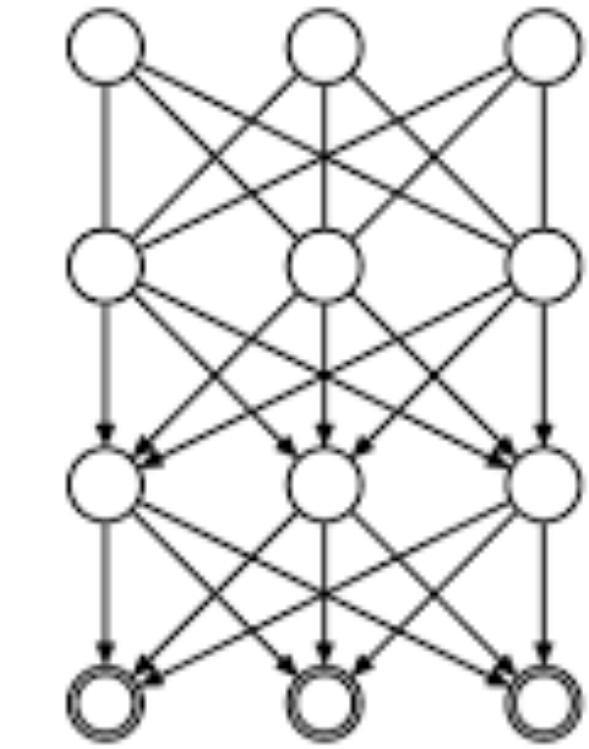
Han, Wang, Fan, LW, Zhang, PRX18'  
Chen, Cheng, Xie, LW, Xiang, PRB 18'  
Cheng, Chen, LW, Entropy 18'+unpublished

# TNSAA: Live long and prosper



**Quantum circuits  
architecture and parametrization**

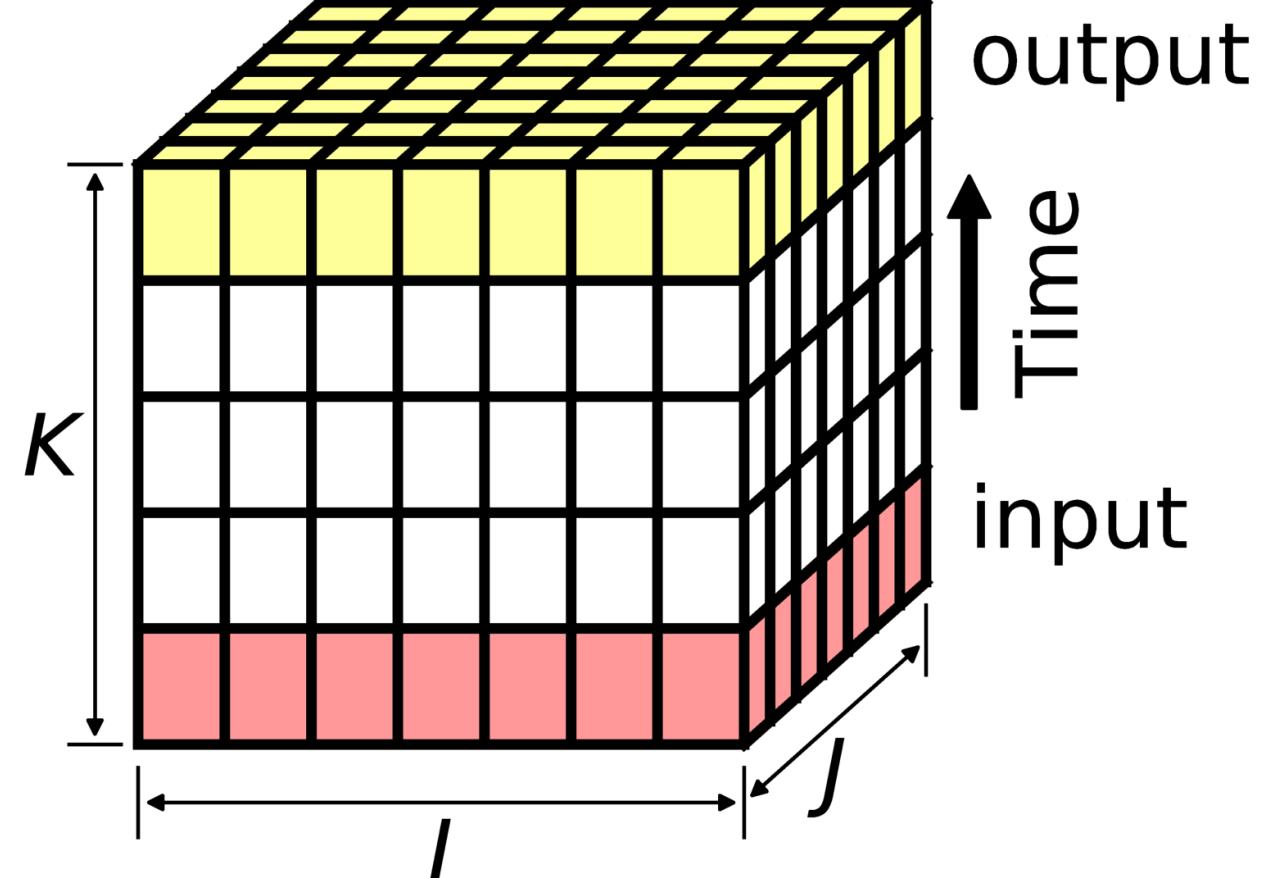
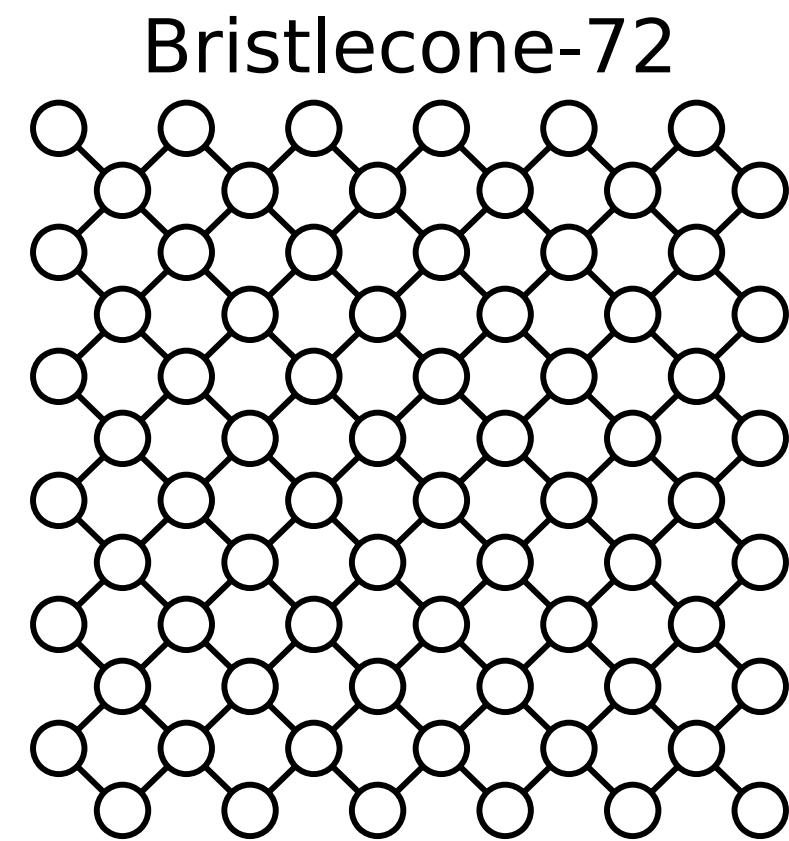
**Neural networks and  
Probabilistic graphical models**



# TNSAA in the era of quantum computing

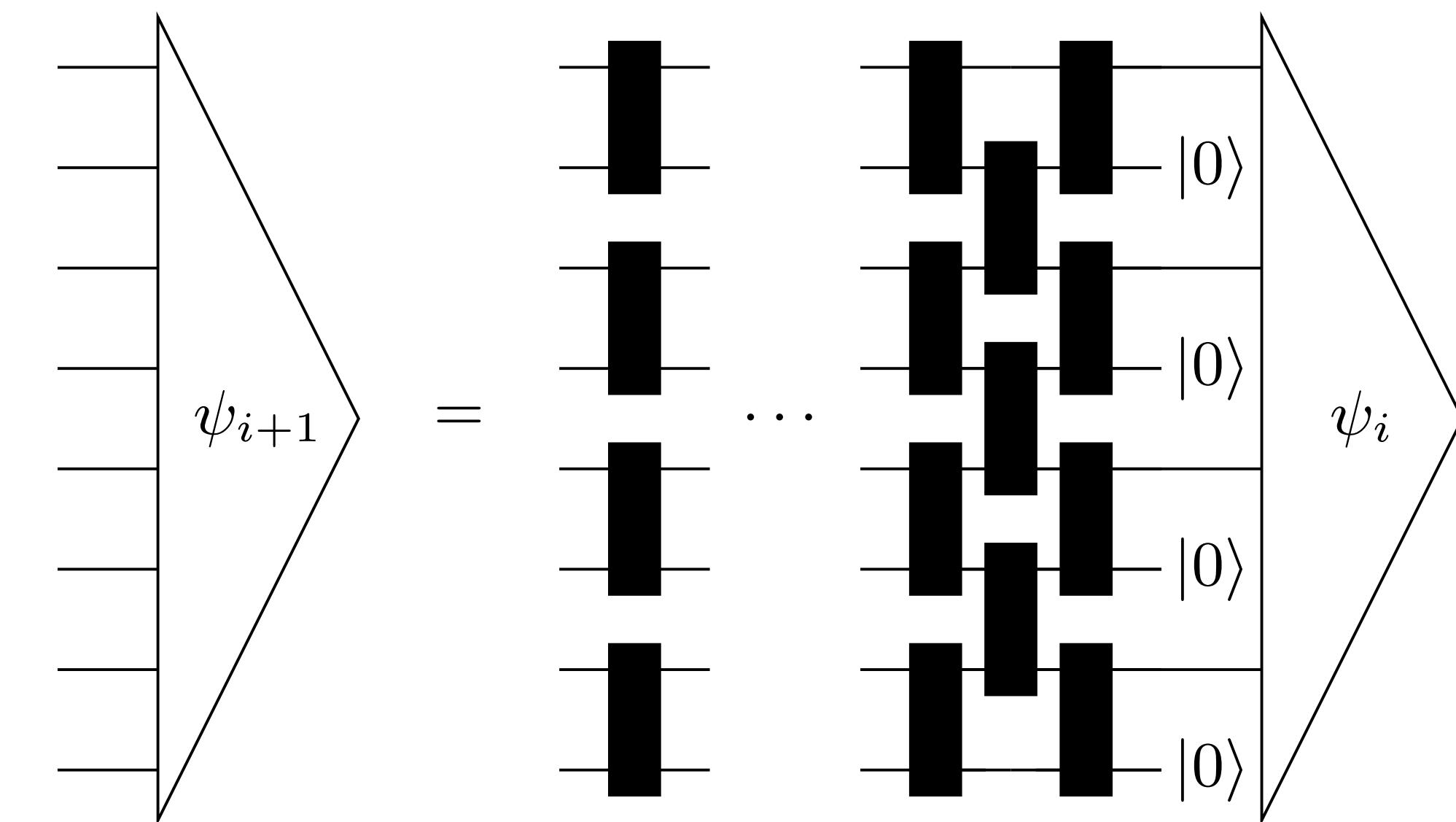
## Simulation and validation of quantum circuits

Villalonga et al, 1811.09599



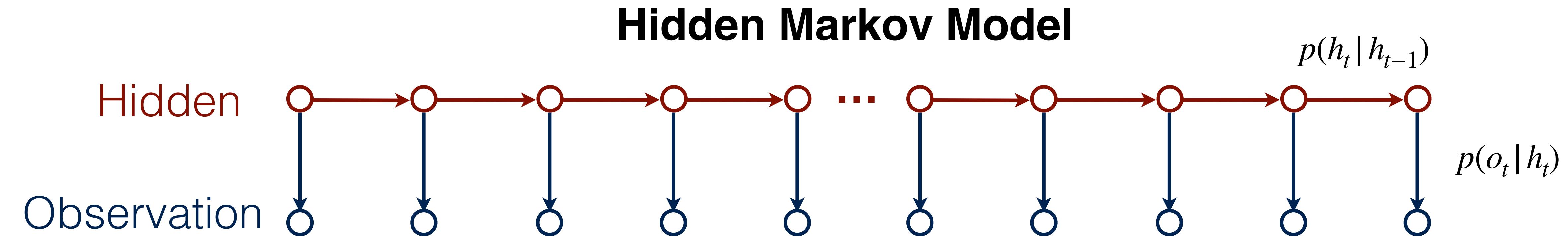
## Noise-resilient quantum circuits with TNS architecture

Kim and Swingle, 1711.07500



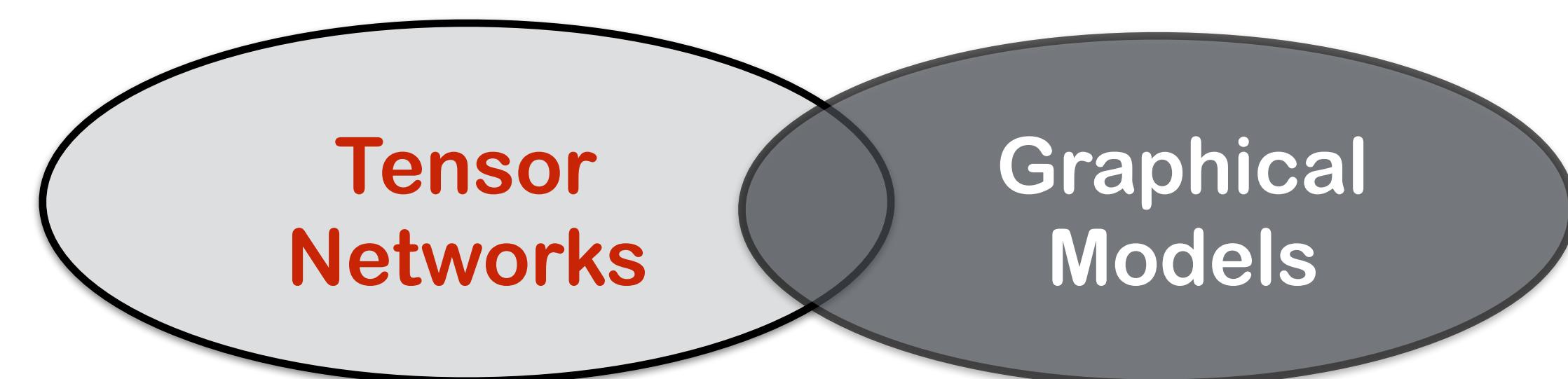
+ quantum tomography, quantum annealing, quantum error correction, holographic duality, and more

# TNSAA in the era of deep learning



- Widely used in speech recognition, bioinformatics, cryptography...
- Learning, inference, and sampling algorithms (Forward-backward message passing, Viterbi, Baum-Welch) => **Think of positive MPS**

Temme, Verstraete, 1003.2545  
Critch, Morton, 1210.2812



*Unleash the power of tensor network states  
and algorithms for generative modeling*

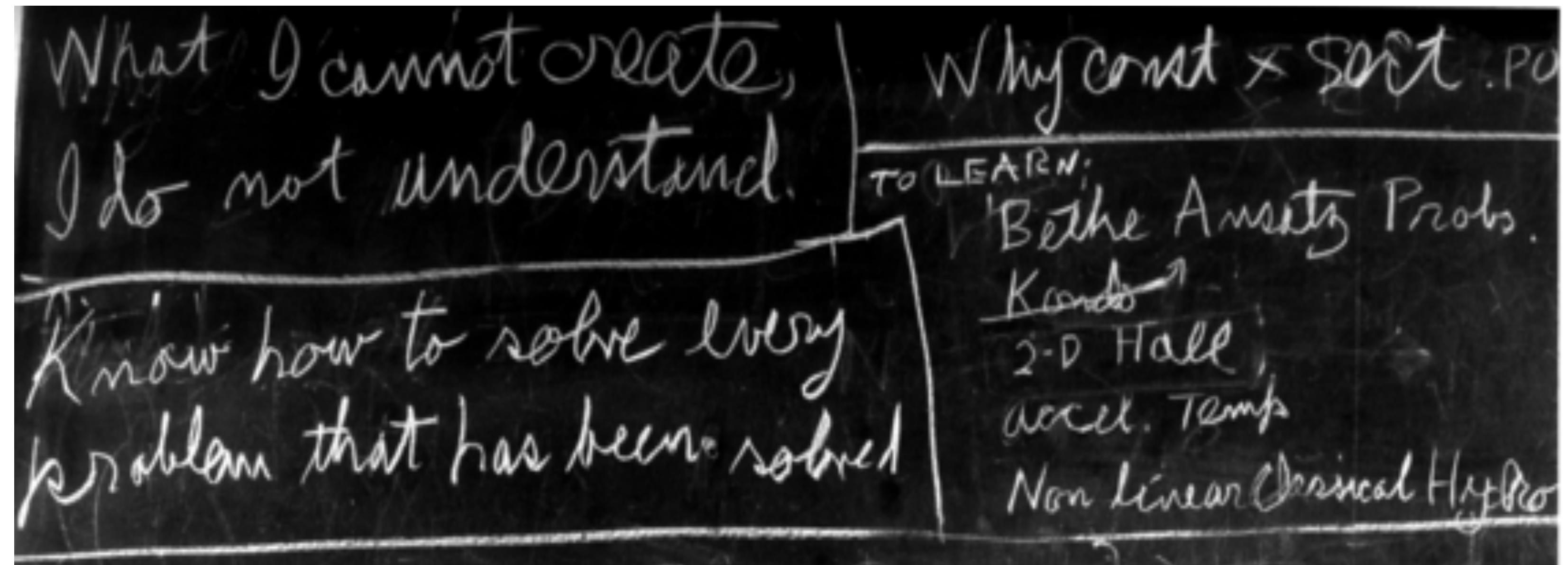
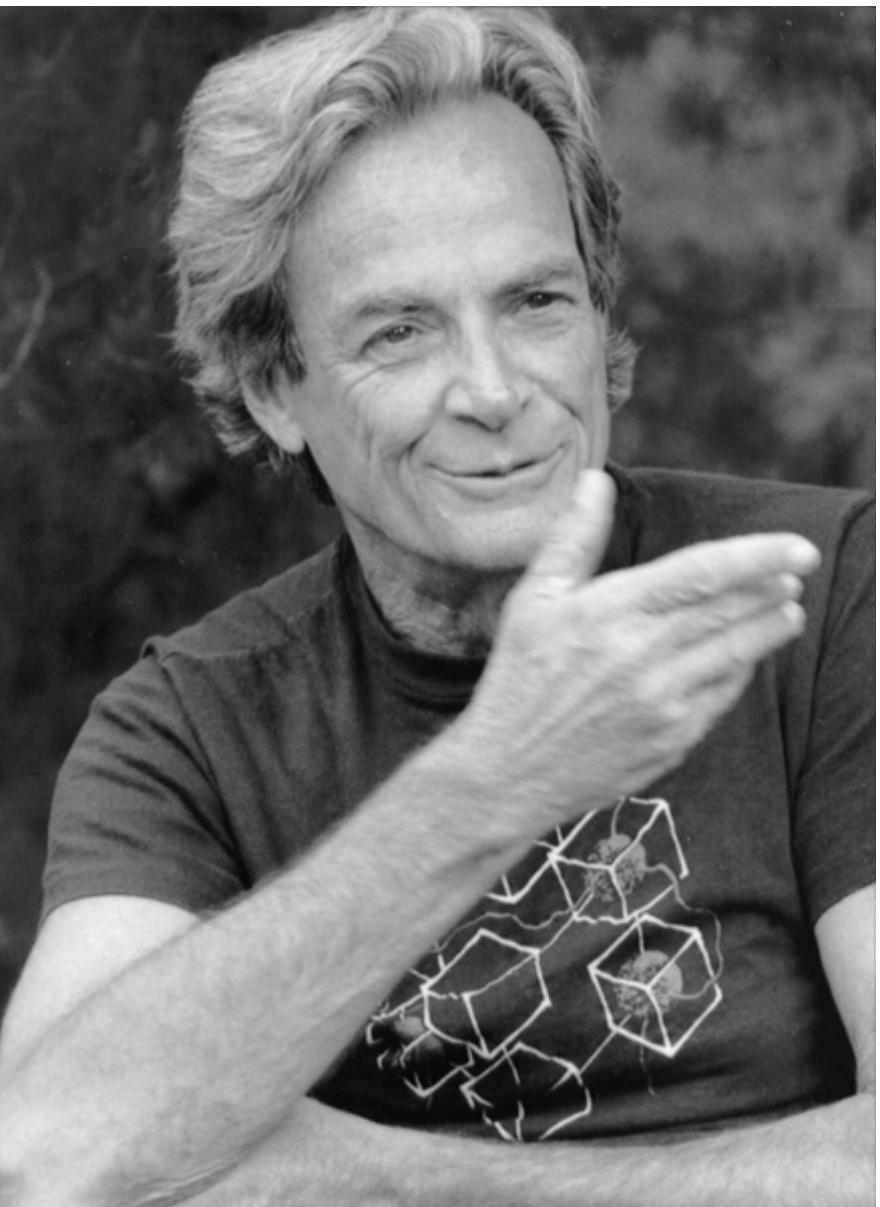
# Discriminative and generative learning


$$y = f(x)$$

or  $p(y | x)$

$$p(x, y)$$

# Discriminative and generative learning



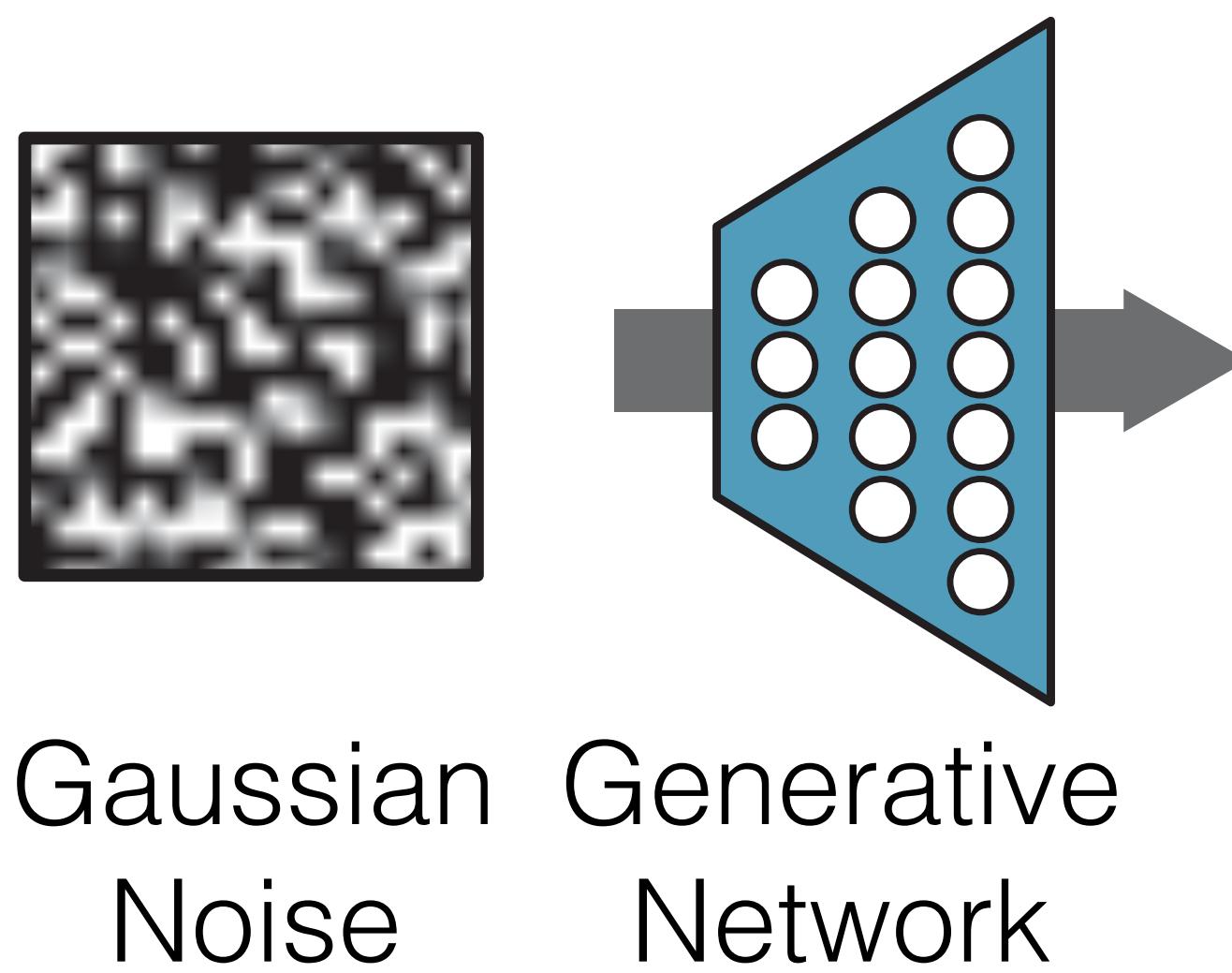
“What I can not create, I do not understand”

# Generated Arts

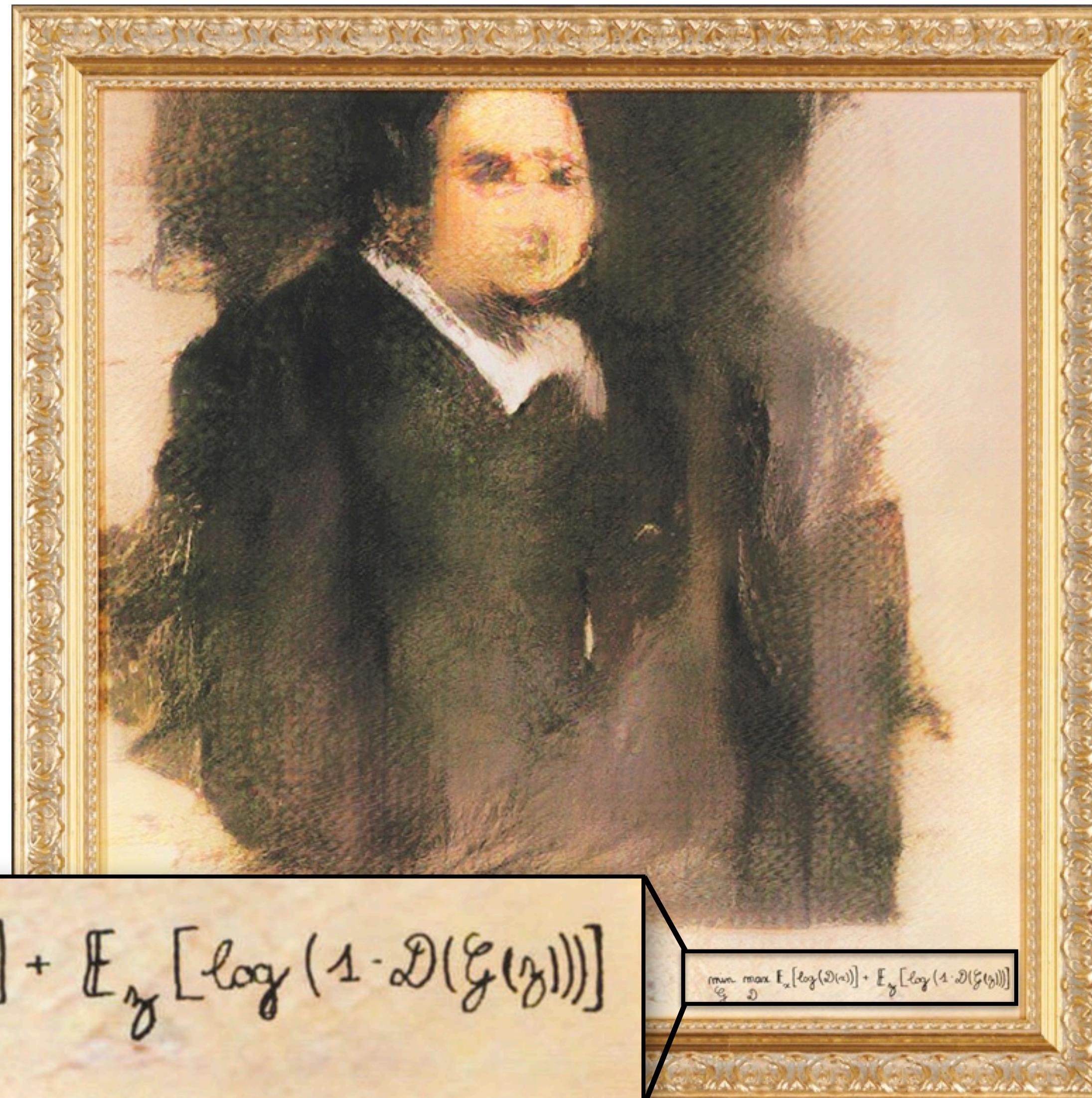


**\$432,500**  
**25 October 2018**  
**Christie's New York**

# Generated Arts

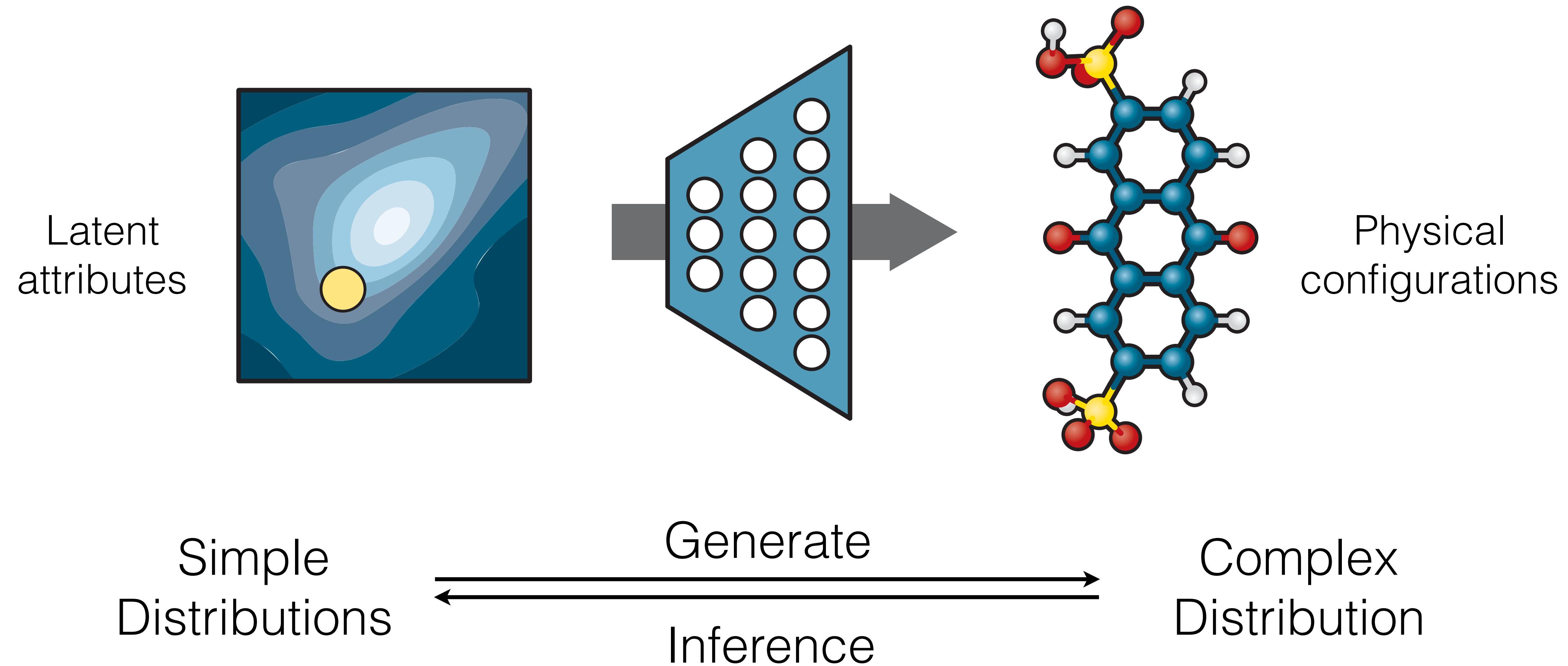


Gaussian Noise      Generative Network



\$432,500  
25 October 2018  
Christie's New York

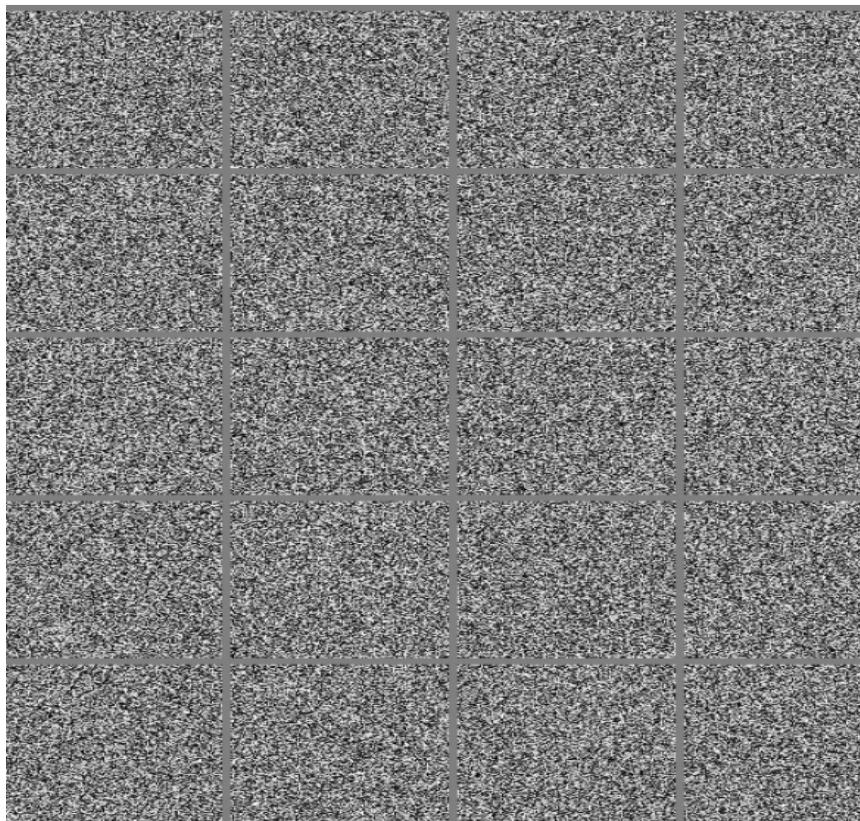
# Generate Molecules



# Probabilistic Generative Modeling

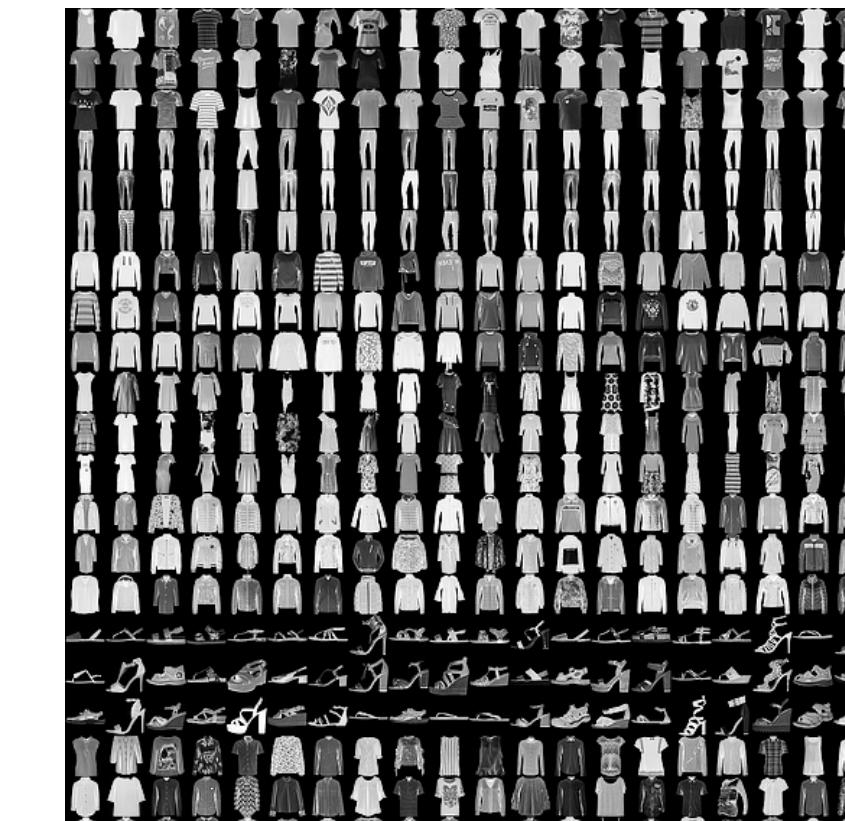
$$p(x)$$

How to express, learn, and sample from a high-dimensional probability distribution ?



“random” images

8	9	0	1	2	3	4	7	8	9	0	1	2	3	4	5	6	7	8	6
4	2	6	4	7	5	5	4	7	8	9	2	9	3	9	3	8	2	0	5
0	1	0	4	2	6	5	3	5	3	8	0	0	3	4	1	5	3	0	8
3	0	6	2	7	1	1	8	1	7	1	3	8	9	7	6	7	4	1	6
7	5	1	7	1	9	8	0	6	9	4	9	9	3	7	1	9	2	2	5
3	7	8	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	0
1	2	3	4	5	6	7	8	9	8	1	0	5	5	1	9	0	4	1	9
3	8	4	7	7	8	5	0	6	5	5	3	3	3	9	8	1	4	0	6
1	0	0	6	2	1	1	3	2	8	8	7	8	4	6	0	2	0	3	6
8	7	1	5	9	9	3	2	4	9	4	4	5	3	2	8	5	9	4	1
6	5	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7
8	9	0	1	2	3	4	5	6	7	8	9	6	4	2	6	4	7	5	5
4	7	8	9	2	9	3	9	3	8	2	0	9	8	0	5	6	0	1	0
4	2	6	5	5	5	4	3	4	1	5	3	0	8	3	0	6	2	7	1
1	8	1	7	1	3	8	5	4	2	0	9	7	6	7	4	1	6	8	4
7	5	1	2	6	7	1	9	8	0	6	9	4	9	9	6	2	3	7	1
9	2	2	5	3	7	8	0	1	2	3	4	5	6	7	8	0	1	2	3
4	5	6	7	8	0	1	2	3	4	5	6	7	8	9	2	1	2	1	3
9	9	8	5	3	7	0	7	7	5	7	9	9	4	7	0	3	4	1	4
4	7	5	8	1	4	8	4	1	8	6	4	6	3	5	7	2	5	9	



“natural” images

# Probabilistic modeling

How to  
high-d

from a  
solution ?

Page 159

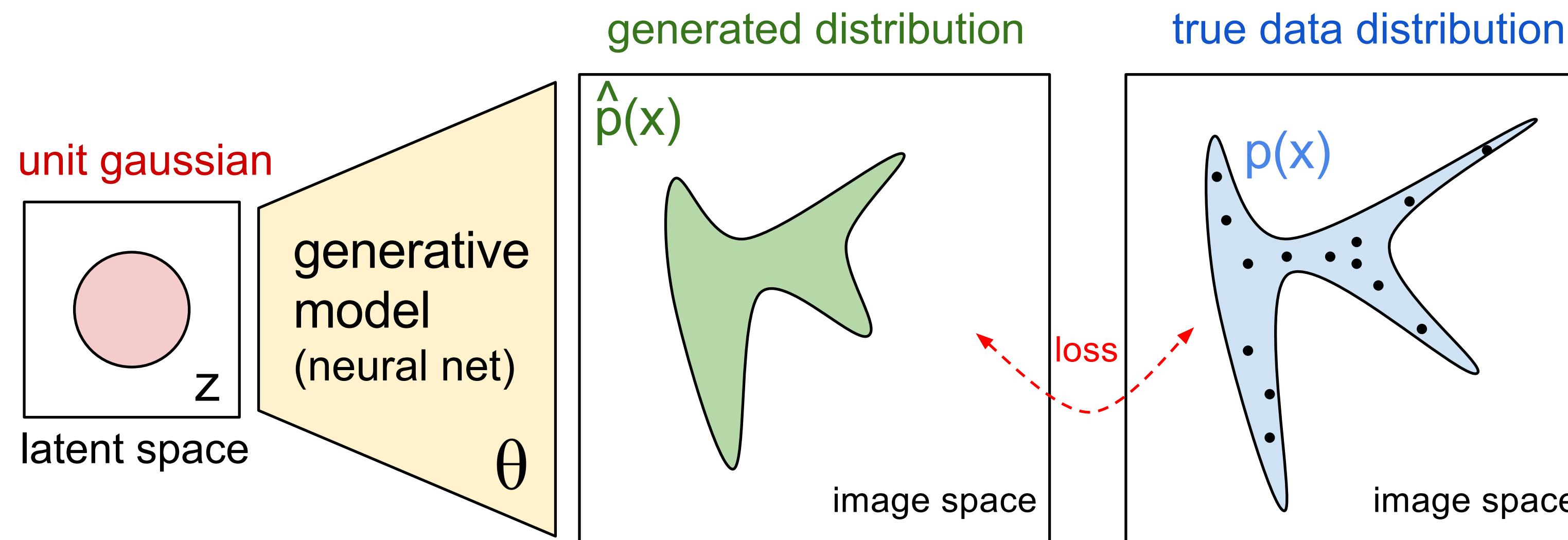
*“... the images encountered in  
AI applications occupy a  
negligible proportion of  
the volume of image space.”*

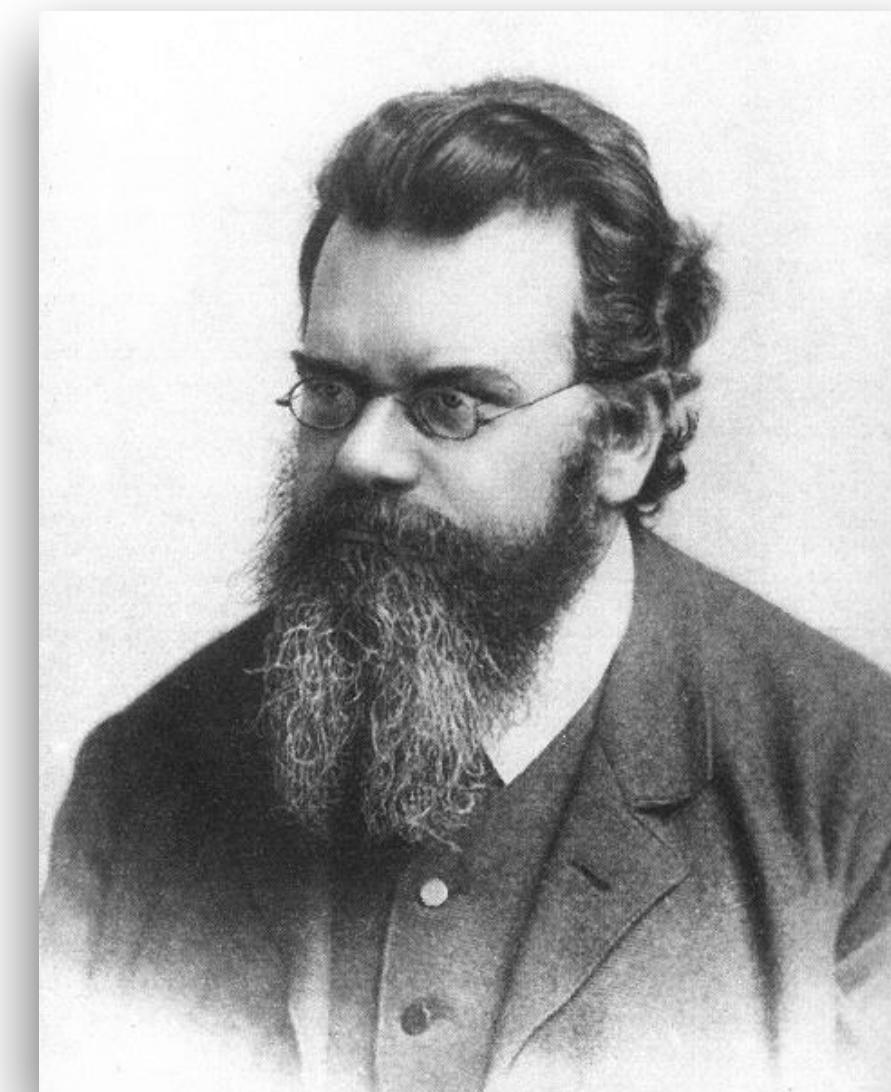
“random”

# Probabilistic Generative Modeling

$$p(x)$$

How to express, learn, and sample from a high-dimensional probability distribution ?





# *Boltzmann Machines*

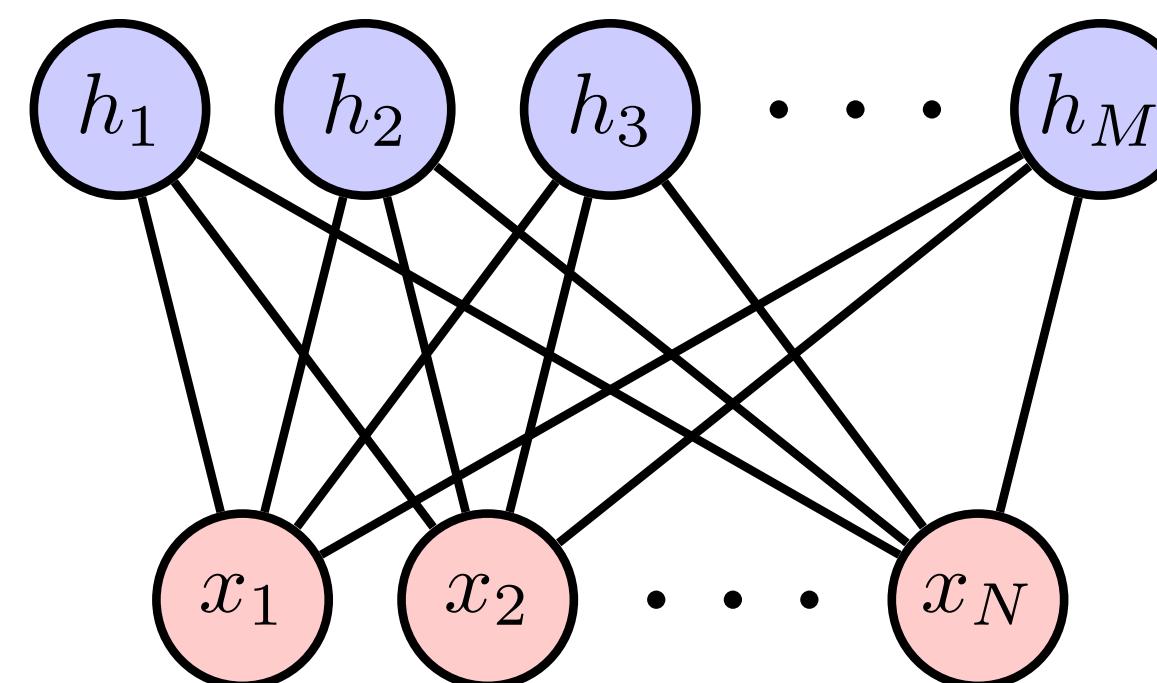
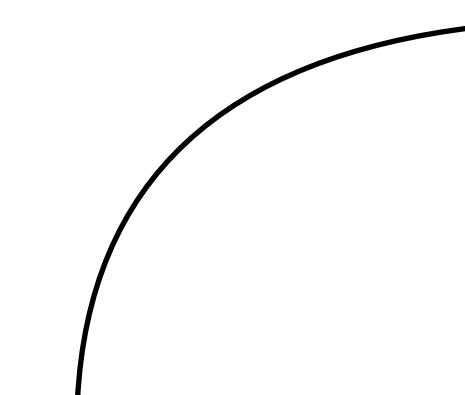
$$p(x) = \frac{e^{-E(x)}}{Z}$$

statistical physics

# Generative Modeling using Boltzmann Machines

Negative log-likelihood loss  $\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ln p(x)$

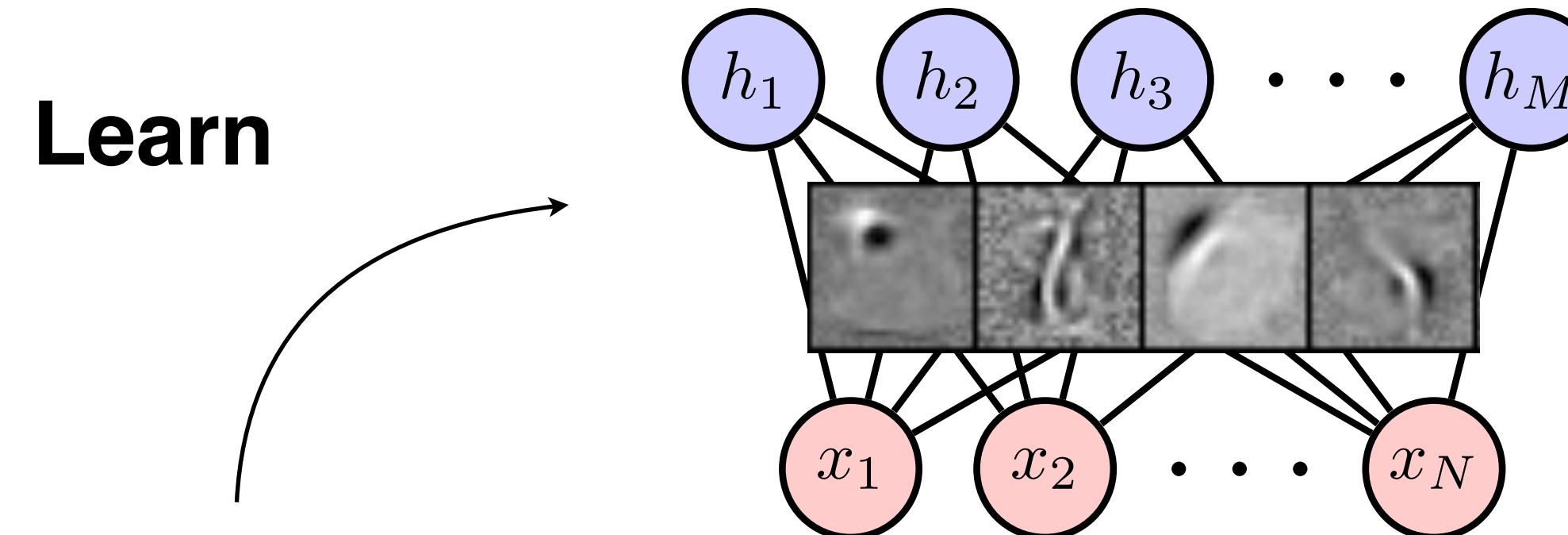
**Learn**



6	2	7	4	2	1	9
1	2	5	3	0	7	5
8	1	8	4	2	6	6
0	7	9	8	6	3	2
7	5	0	5	7	9	5
1	8	7	0	6	5	0
7	5	4	8	4	4	7

# Generative Modeling using Boltzmann Machines

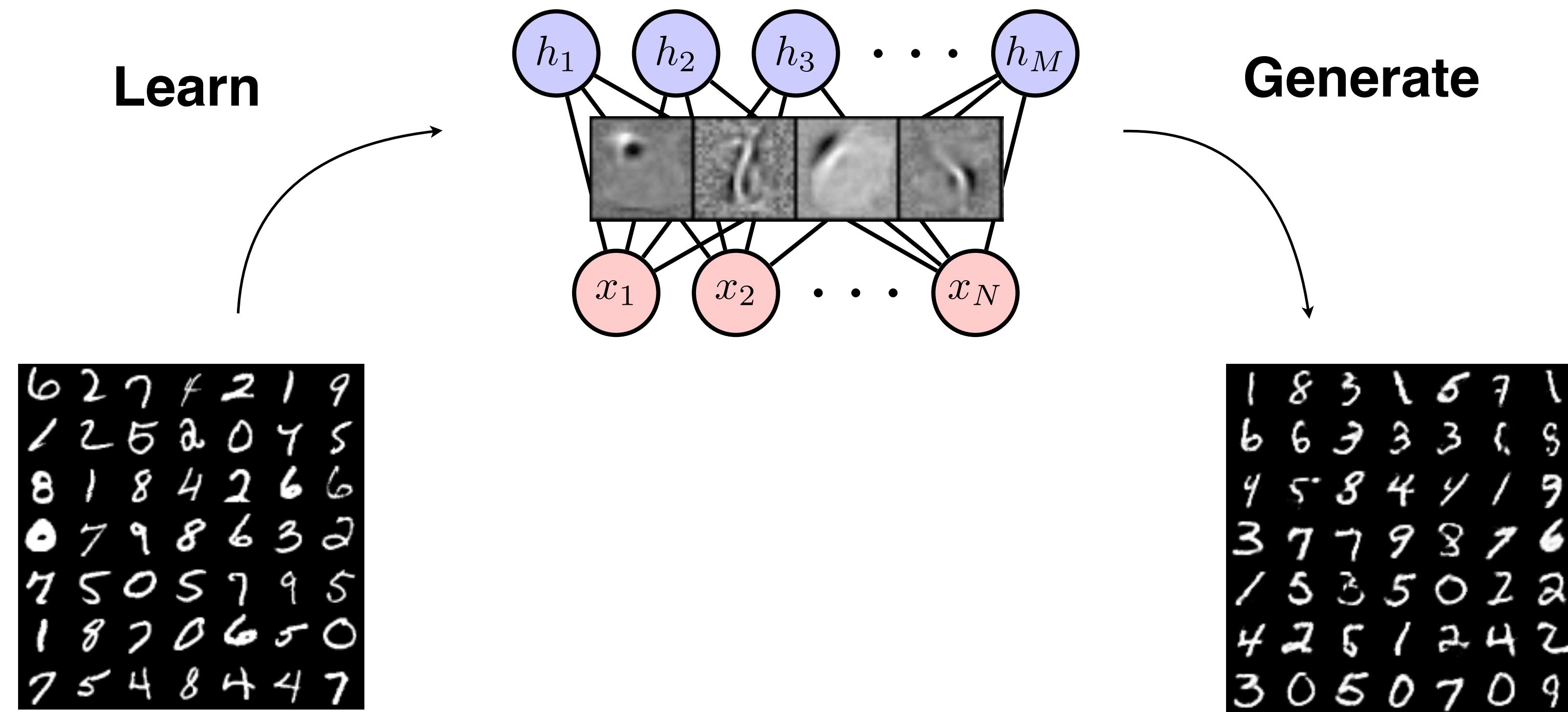
Negative log-likelihood loss  $\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ln p(x)$



6	2	7	4	2	1	9
1	2	5	3	0	7	5
8	1	8	4	2	6	6
0	7	9	8	6	3	2
7	5	0	5	7	9	5
1	8	7	0	6	5	0
7	5	4	8	4	4	7

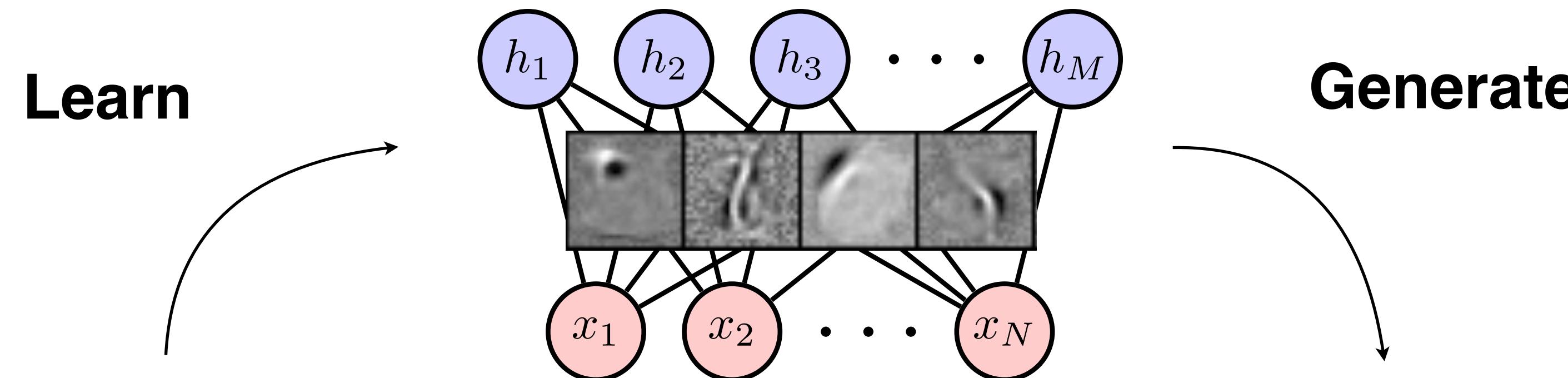
# Generative Modeling using Boltzmann Machines

Negative log-likelihood loss  $\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ln p(x)$



# Generative Modeling using Boltzmann Machines

Negative log-likelihood loss  $\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ln p(x) = \langle E(x) \rangle_{x \sim \mathcal{D}} + \ln Z$

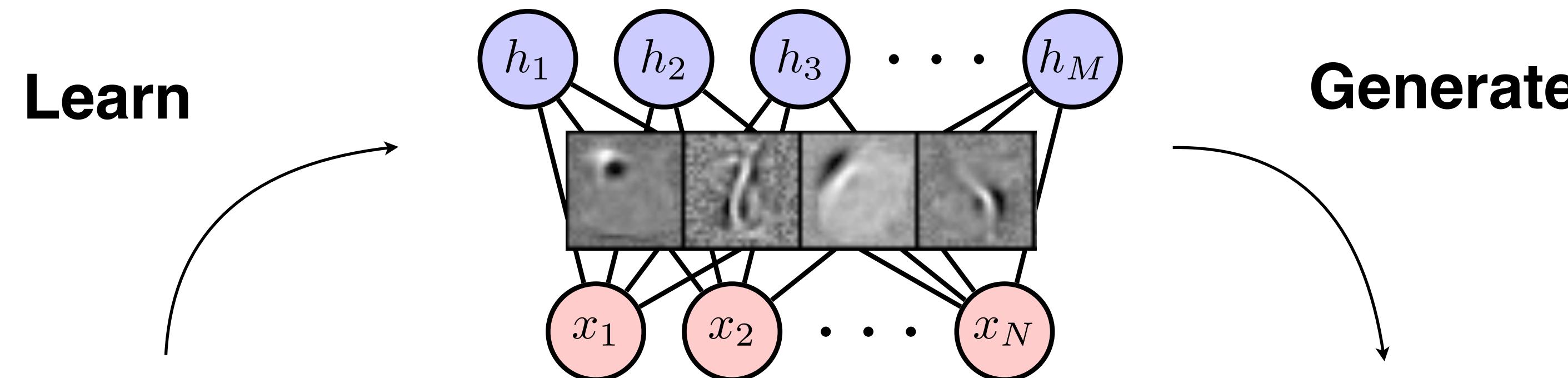


6 2 7 4 2 1 9
1 2 5 2 0 7 5
8 1 8 4 2 6 6
0 7 9 8 6 3 2
7 5 0 5 7 9 5
1 8 7 0 6 5 0
7 5 4 8 4 4 7

1 8 3 1 6 7 1
6 6 3 3 3 6 8
4 5 8 4 4 1 9
3 7 7 9 8 7 6
1 5 3 5 0 2 2
4 2 5 1 2 4 2
3 0 5 0 7 0 9

# Generative Modeling using Boltzmann Machines

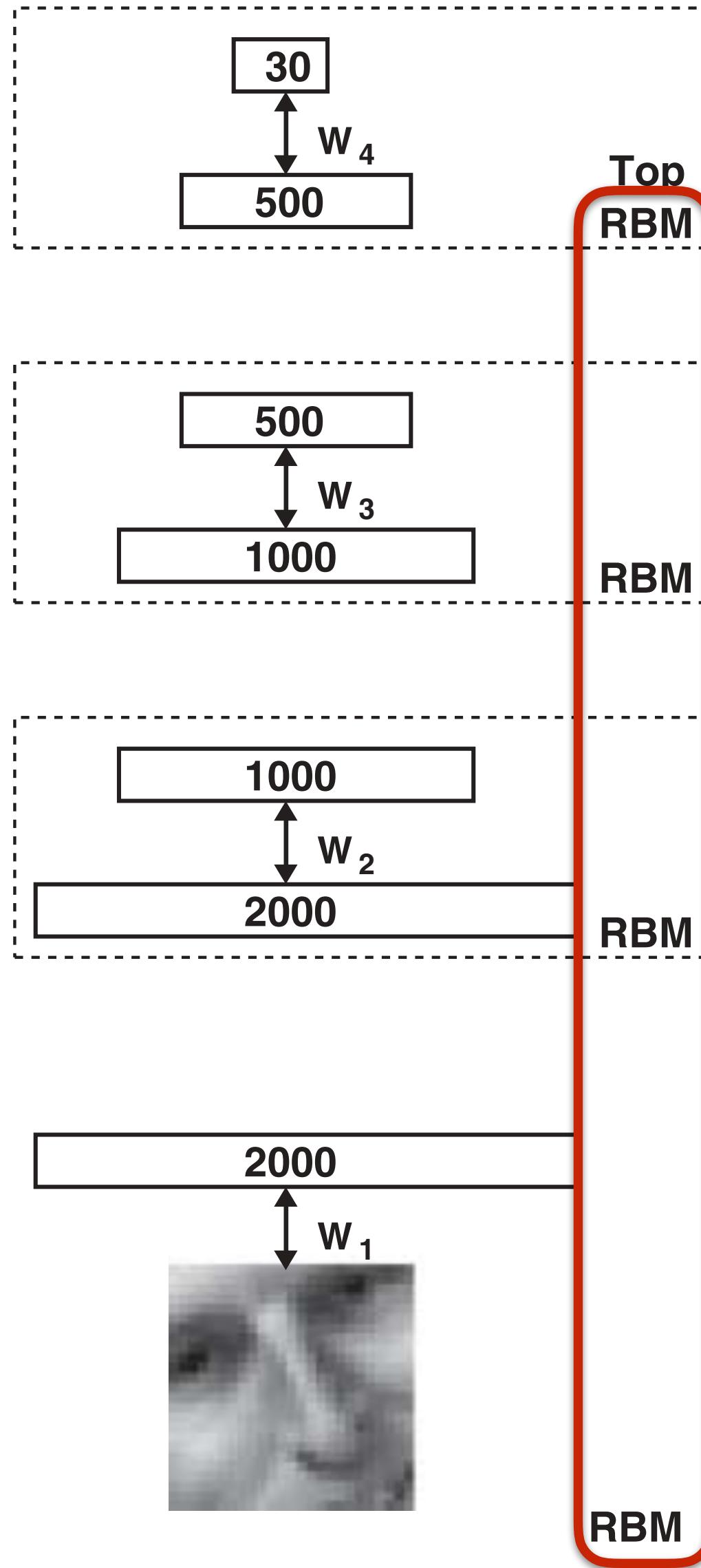
Negative log-likelihood loss  $\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ln p(x) = \langle E(x) \rangle_{x \sim \mathcal{D}} + \ln Z$



6	2	7	4	2	1	9
1	2	5	3	0	7	5
8	1	8	4	2	6	6
0	7	9	8	6	3	2
7	5	0	5	7	9	5
1	8	7	0	6	5	0
7	5	4	8	4	4	7

$$\nabla \mathcal{L} = \langle \nabla E \rangle_{x \sim \mathcal{D}} - \langle \nabla E \rangle_{x \sim p(x)}$$

1	8	3	1	6	7	1
6	6	3	3	3	6	8
4	5	8	4	4	1	9
3	7	7	9	8	7	6
1	5	3	5	0	2	2
4	2	5	1	2	4	2
3	0	5	0	7	0	9



Pretraining

Feedback  
to physics

# Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton\* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such “autoencoder” networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is principal components analysis (PCA), which

finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer “encoder” network

2006 VOL 313 SCIENCE [www.sciencemag.org](http://www.sciencemag.org)

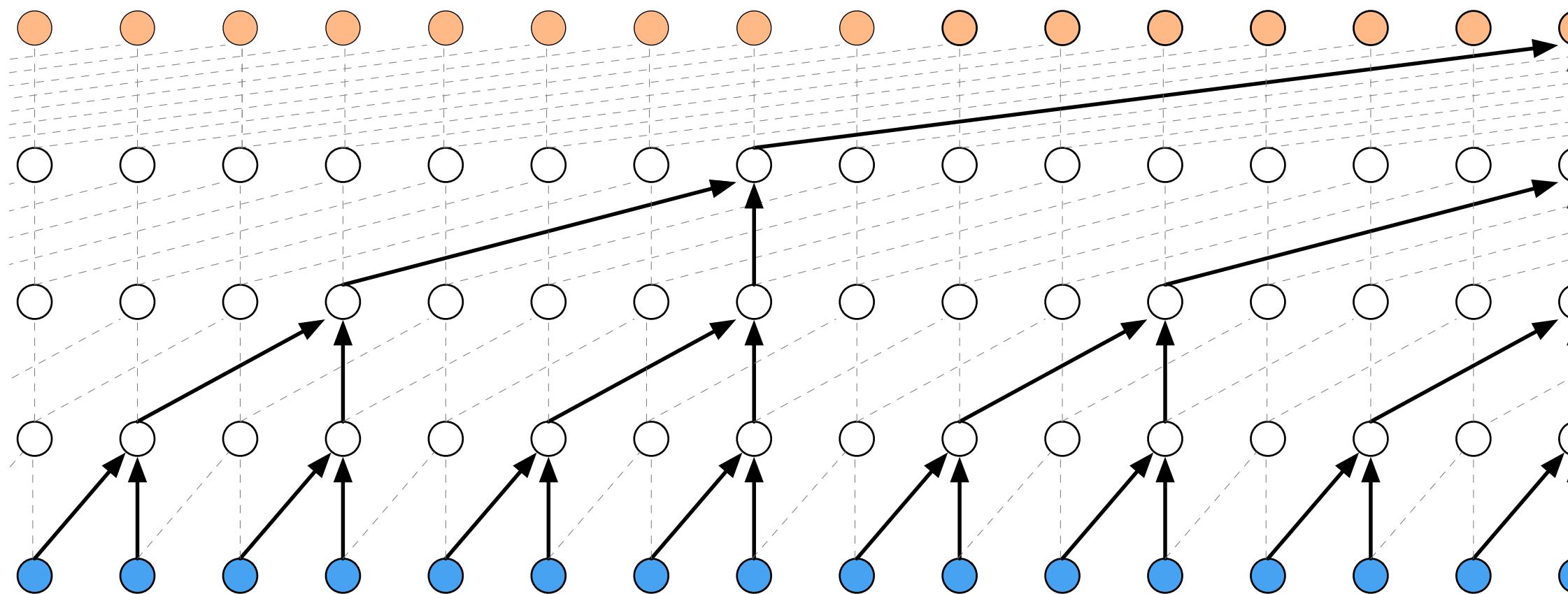
## Renaissance of deep learning

Wavefunctions ansatz  
Quantum state tomography  
Quantum error correction

Monte Carlo updates  
Renormalization group  
..... **see next talk !**

# State-of-the-Art: Autoregressive Models

$$p(\mathbf{x}) = \prod_i p(x_i | \mathbf{x}_{<i})$$



Speech data



WaveNet 1609.03499, 1711.10433

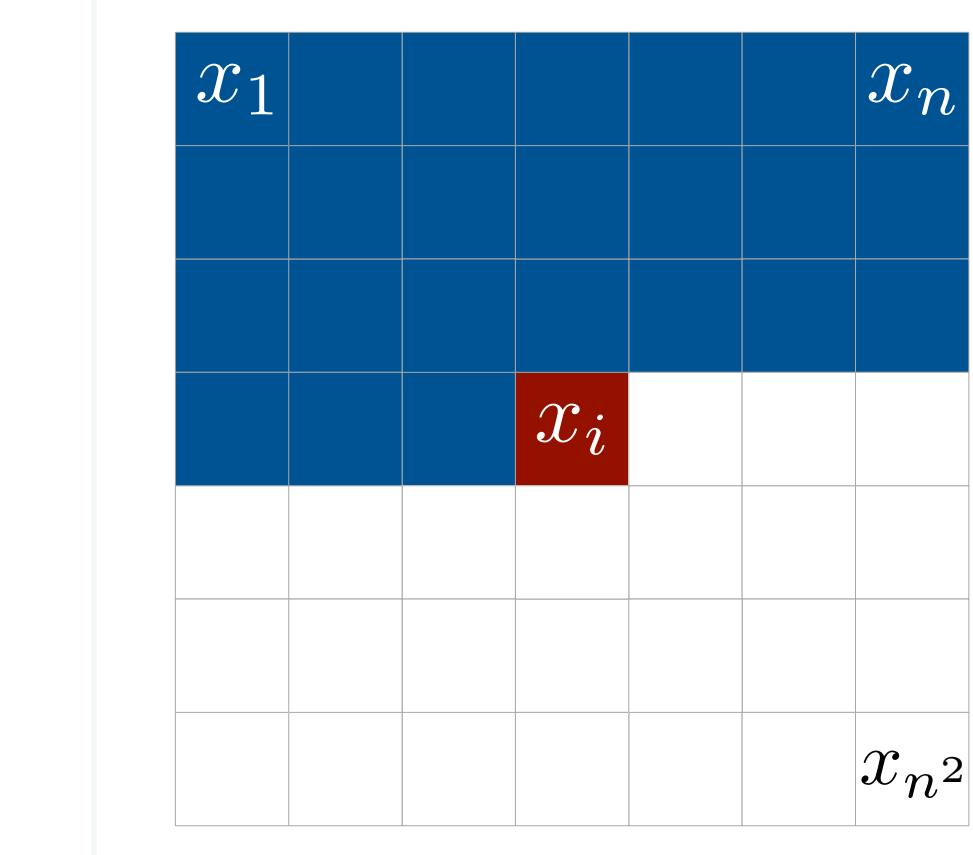
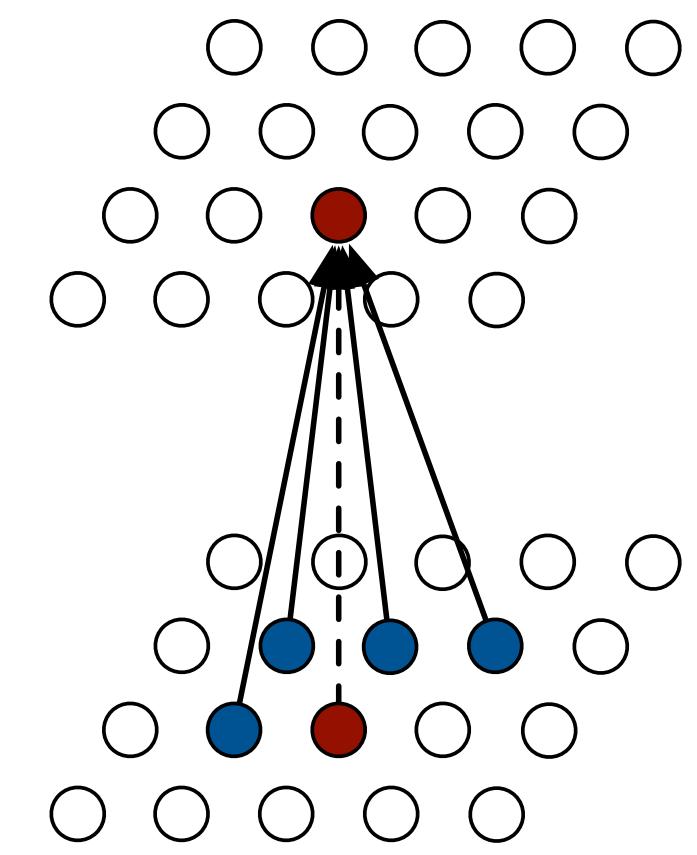


Image data

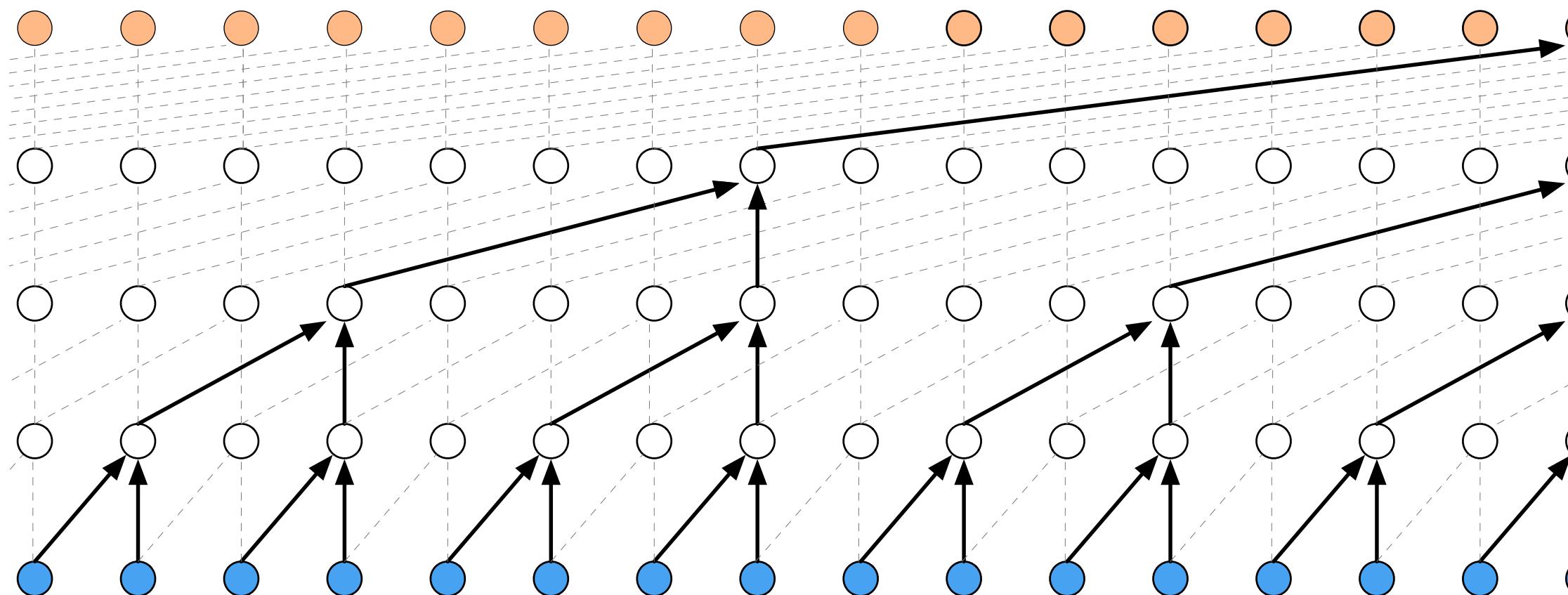


PixelCNN 1601.06759, 1606.05328

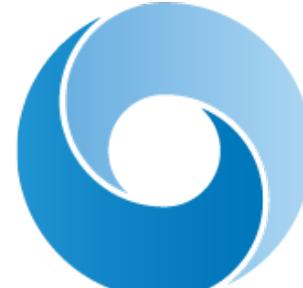


# State-of-the-Art: Autoregressive Models

$$\begin{aligned} p(\mathbf{x}) &= \prod p(x_i | \mathbf{x}_{<i}) \\ &= p(x_1)p(x_2 | x_1)p(x_3 | x_1, x_2)\cdots \end{aligned}$$



Speech data



WaveNet 1609.03499, 1711.10433

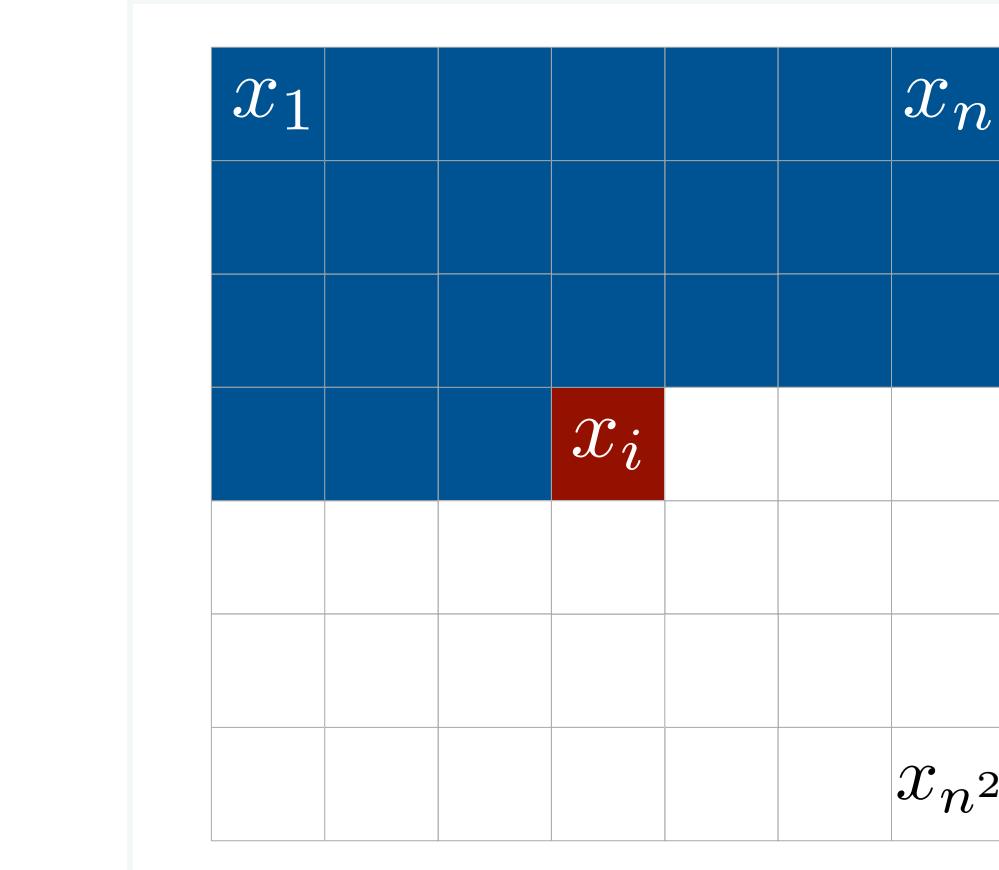
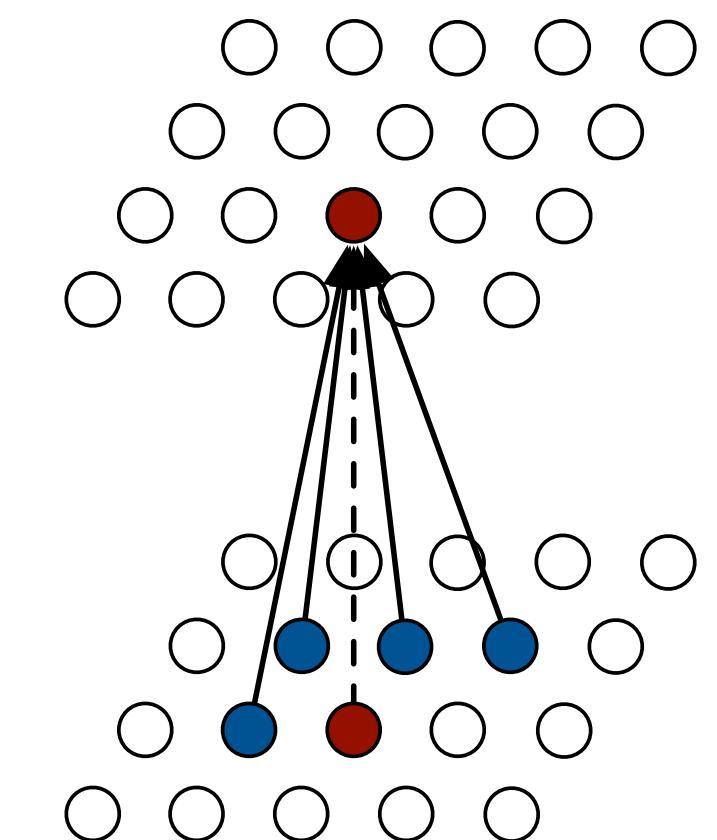


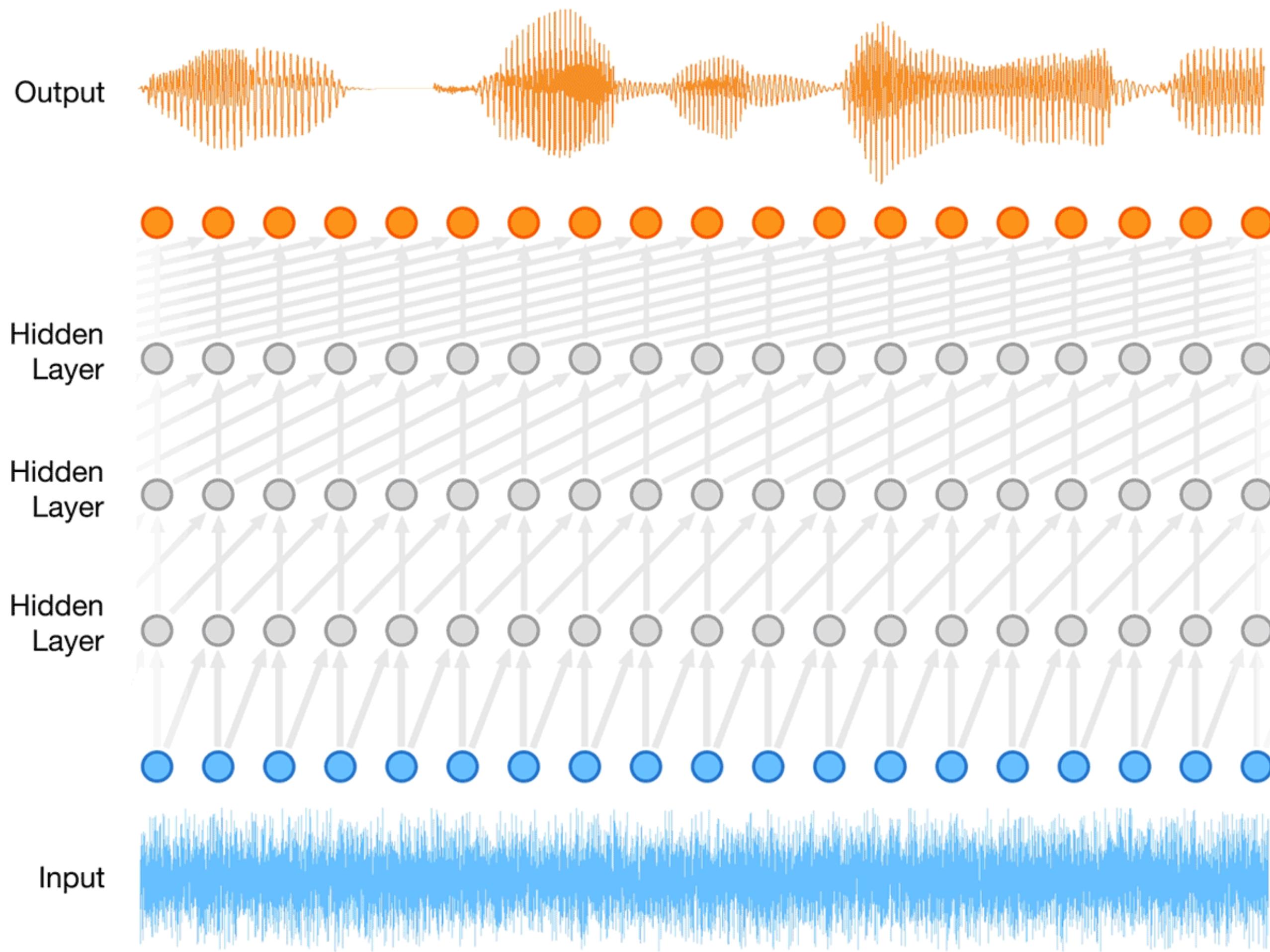
Image data



PixelCNN 1601.06759, 1606.05328



# WaveNet in the Real World



2018 Google I/O

<https://deepmind.com/blog/wavenet-generative-model-raw-audio/>

<https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/>

<https://deepmind.com/blog/wavenet-launches-google-assistant/>

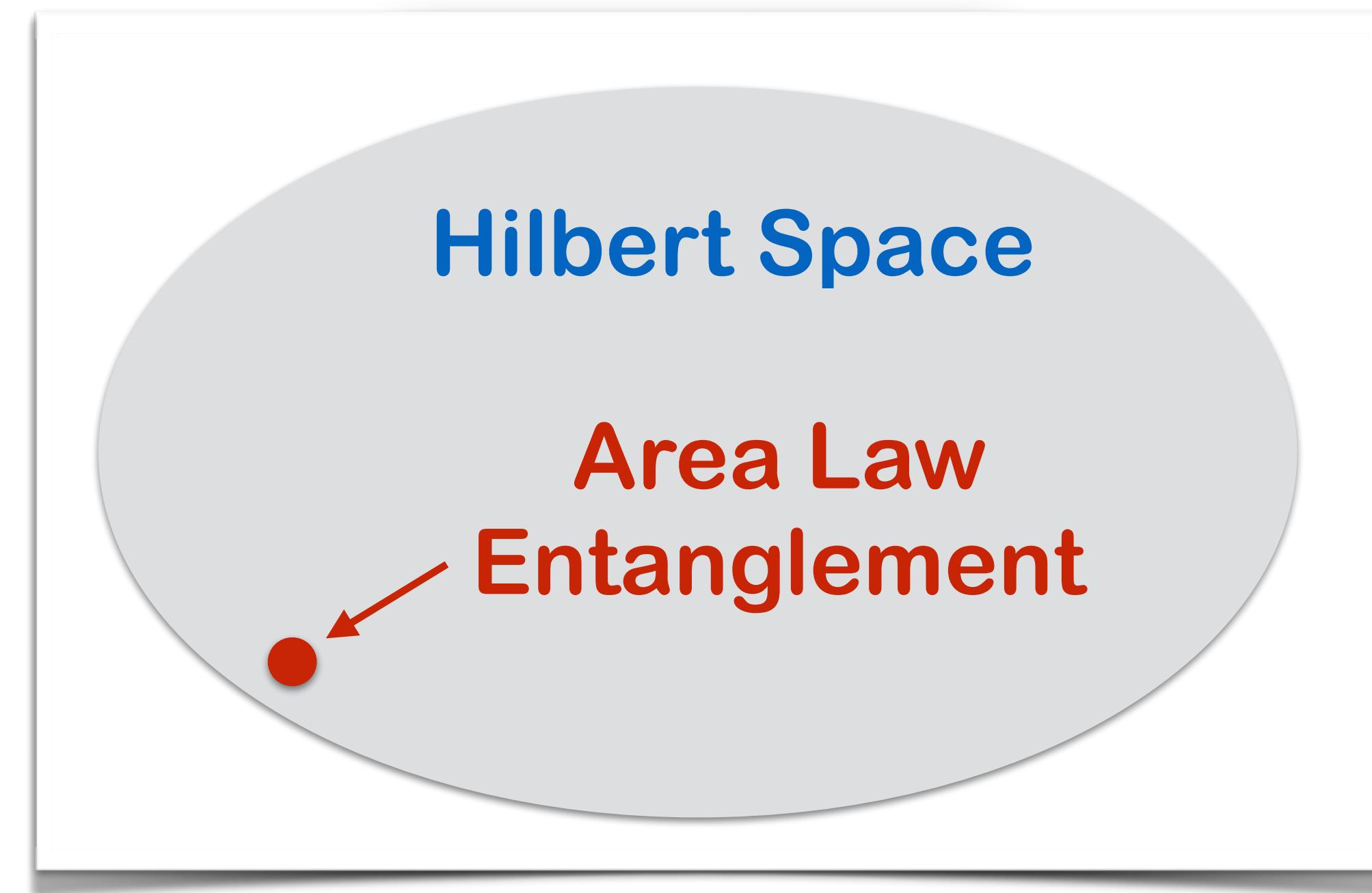




# *Born Machines*

$$p(x) = \frac{|\Psi(x)|^2}{Z}$$

quantum physics

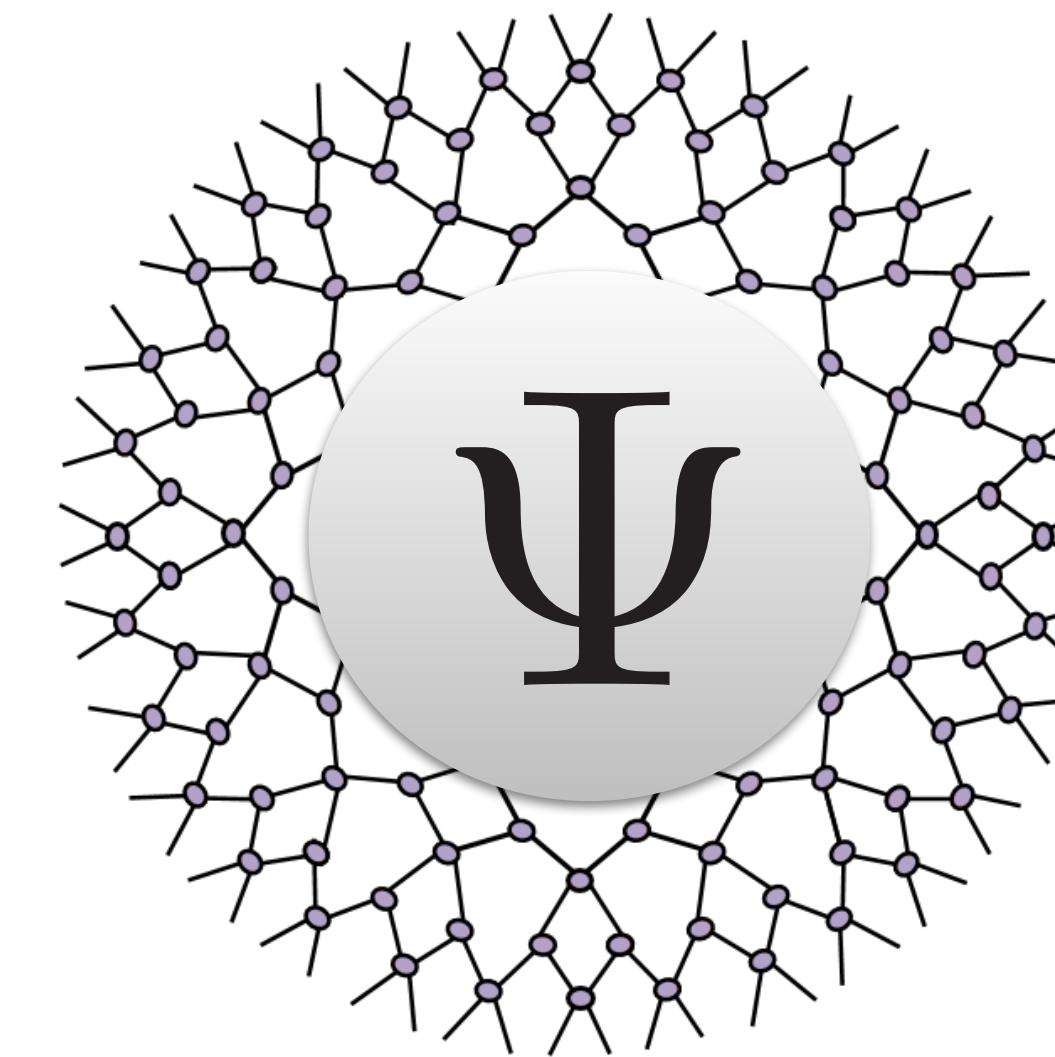


*Born Machines*

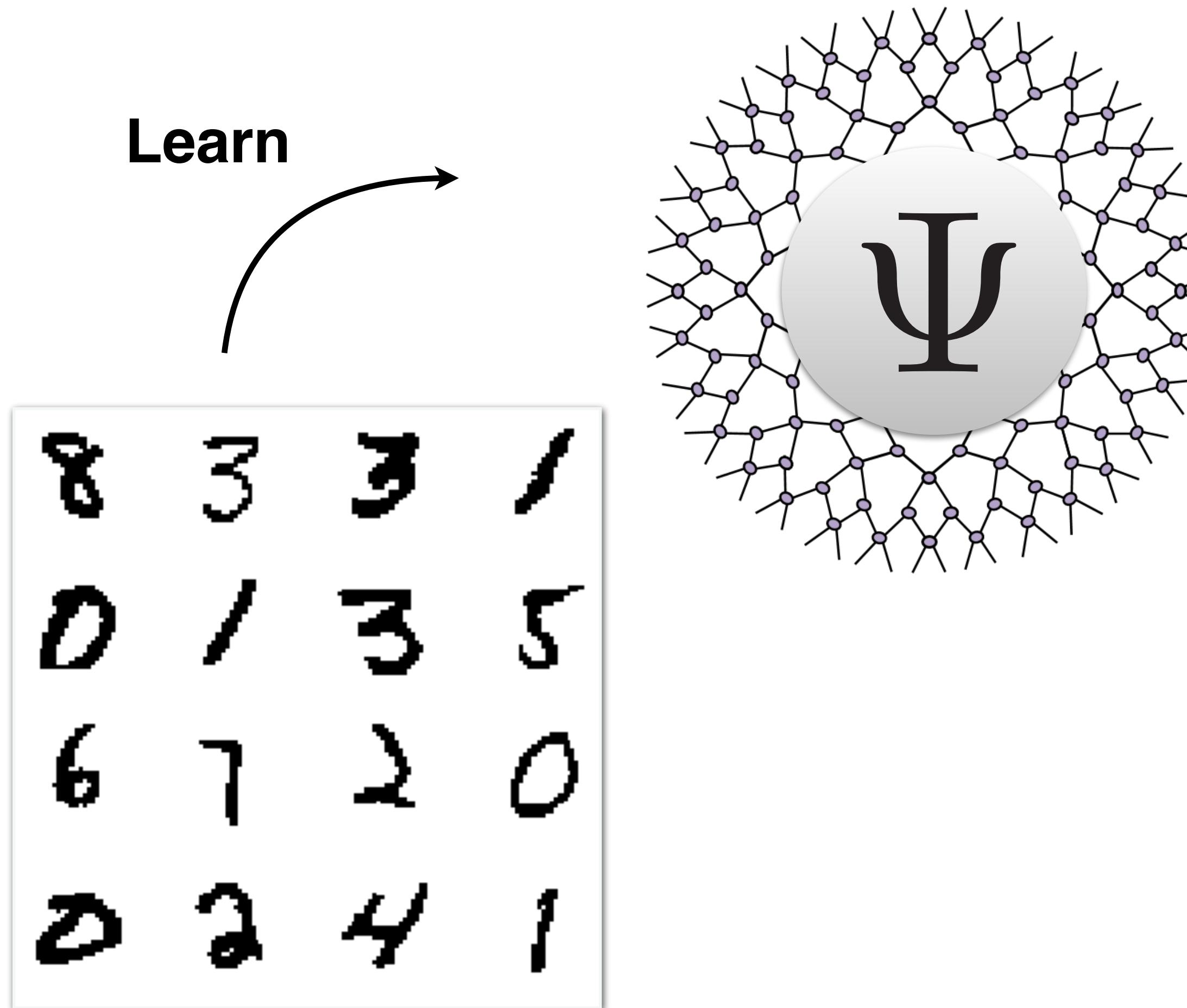
$$p(x) = \frac{|\Psi(x)|^2}{Z}$$

quantum physics

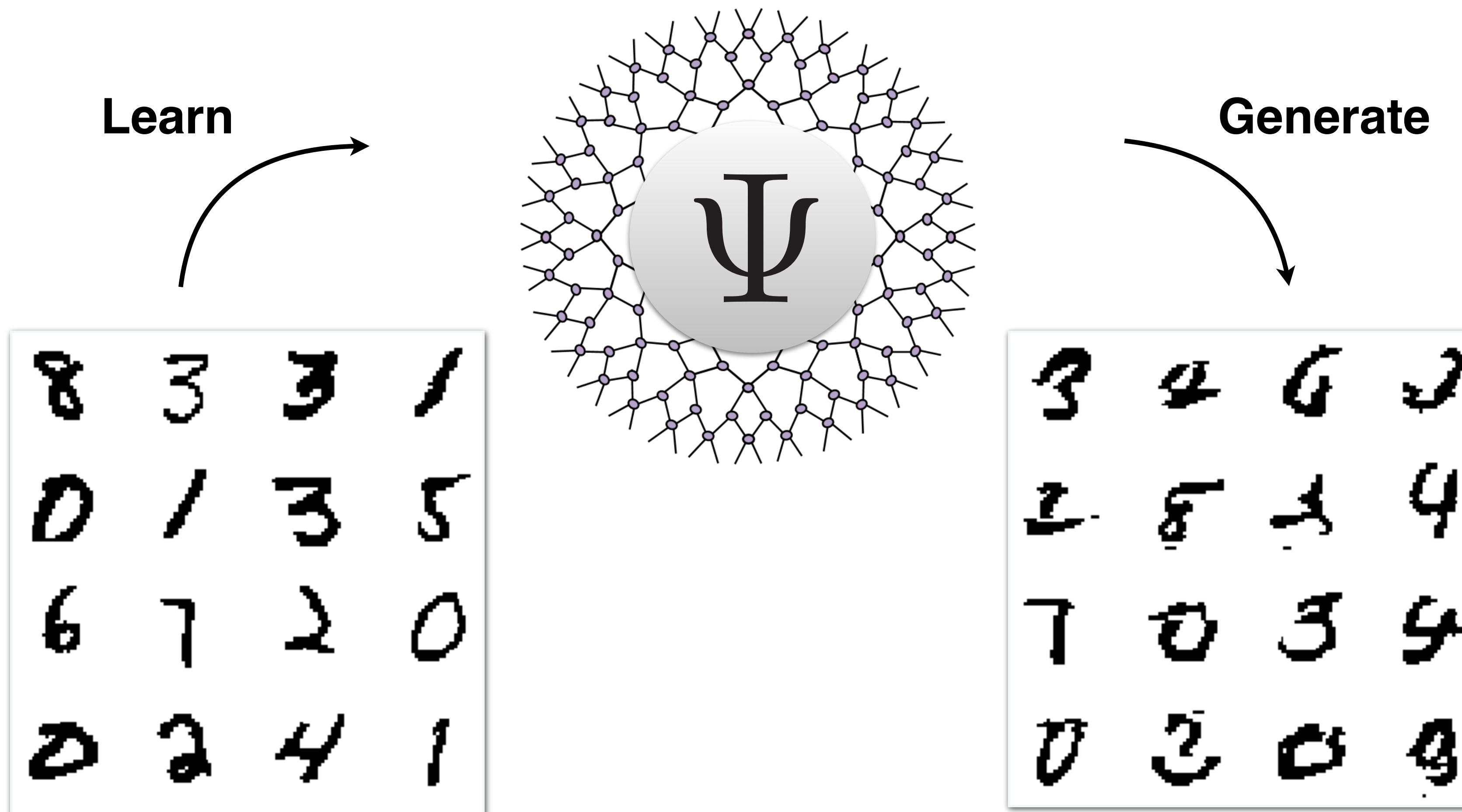
# Quantum inspired generative modeling



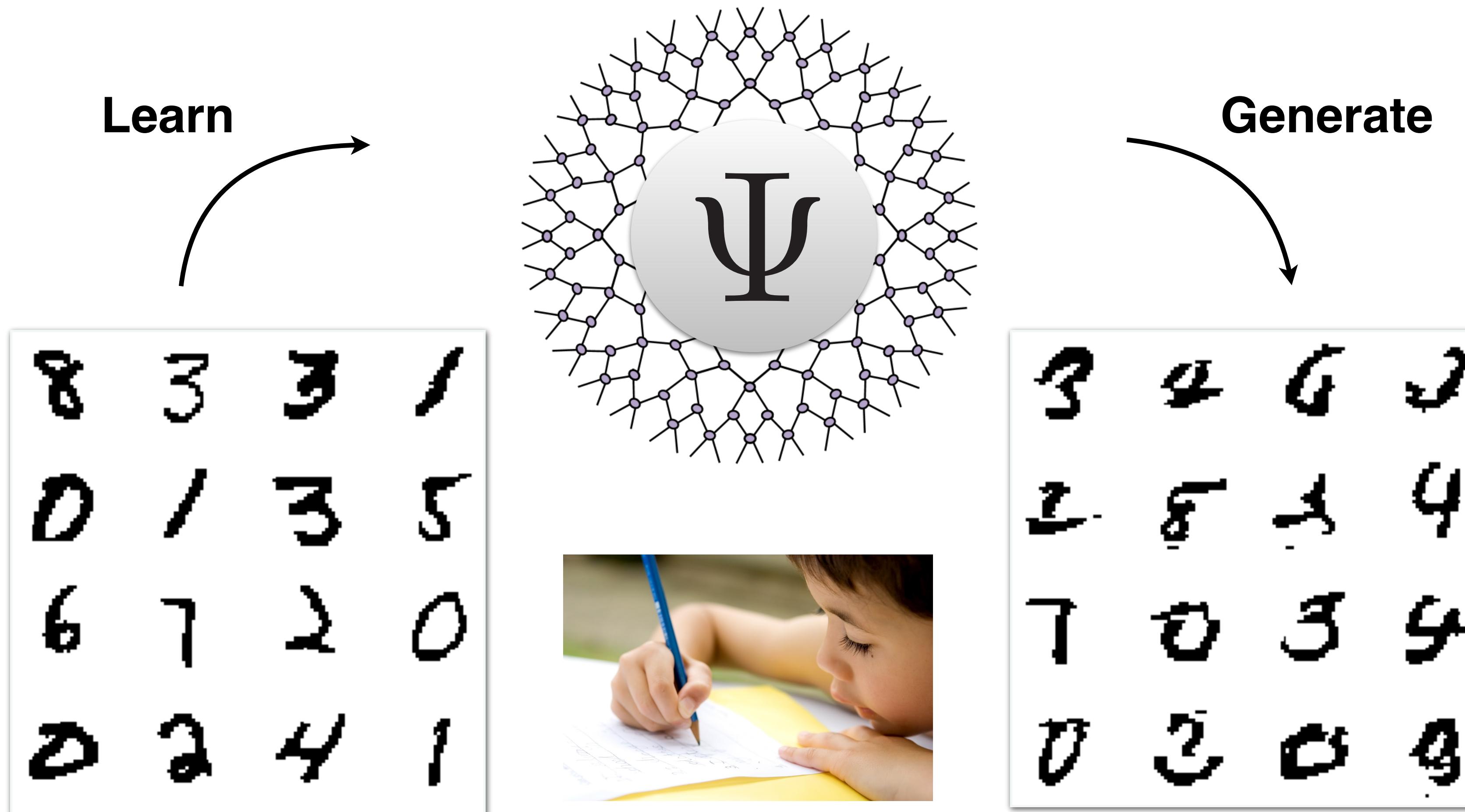
# Quantum inspired generative modeling



# Quantum inspired generative modeling



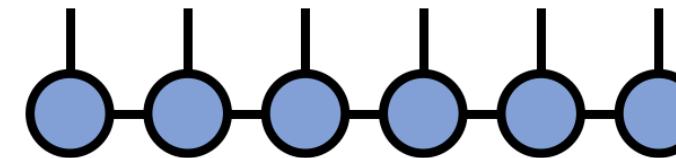
# Quantum inspired generative modeling



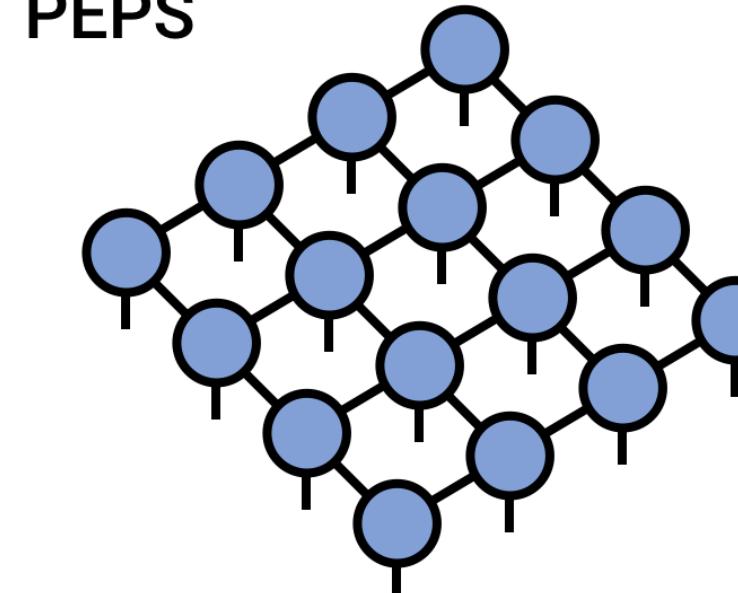
**“Teach a quantum state to write digits”**

# Generative modeling using Tensor Network States

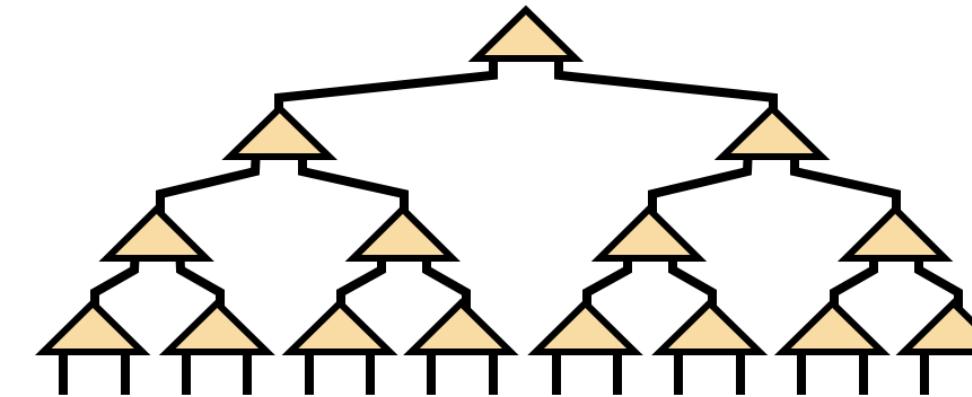
Matrix Product State /  
Tensor Train



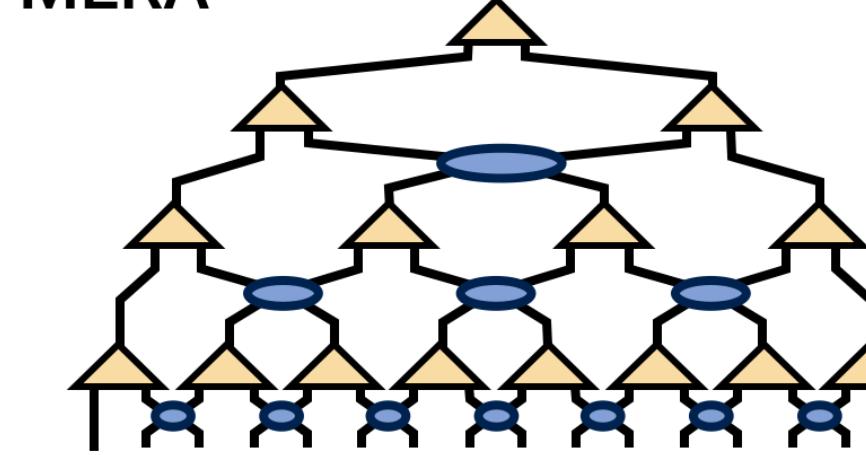
PEPS



Tree Tensor Network /  
Hierarchical Tucker



MERA



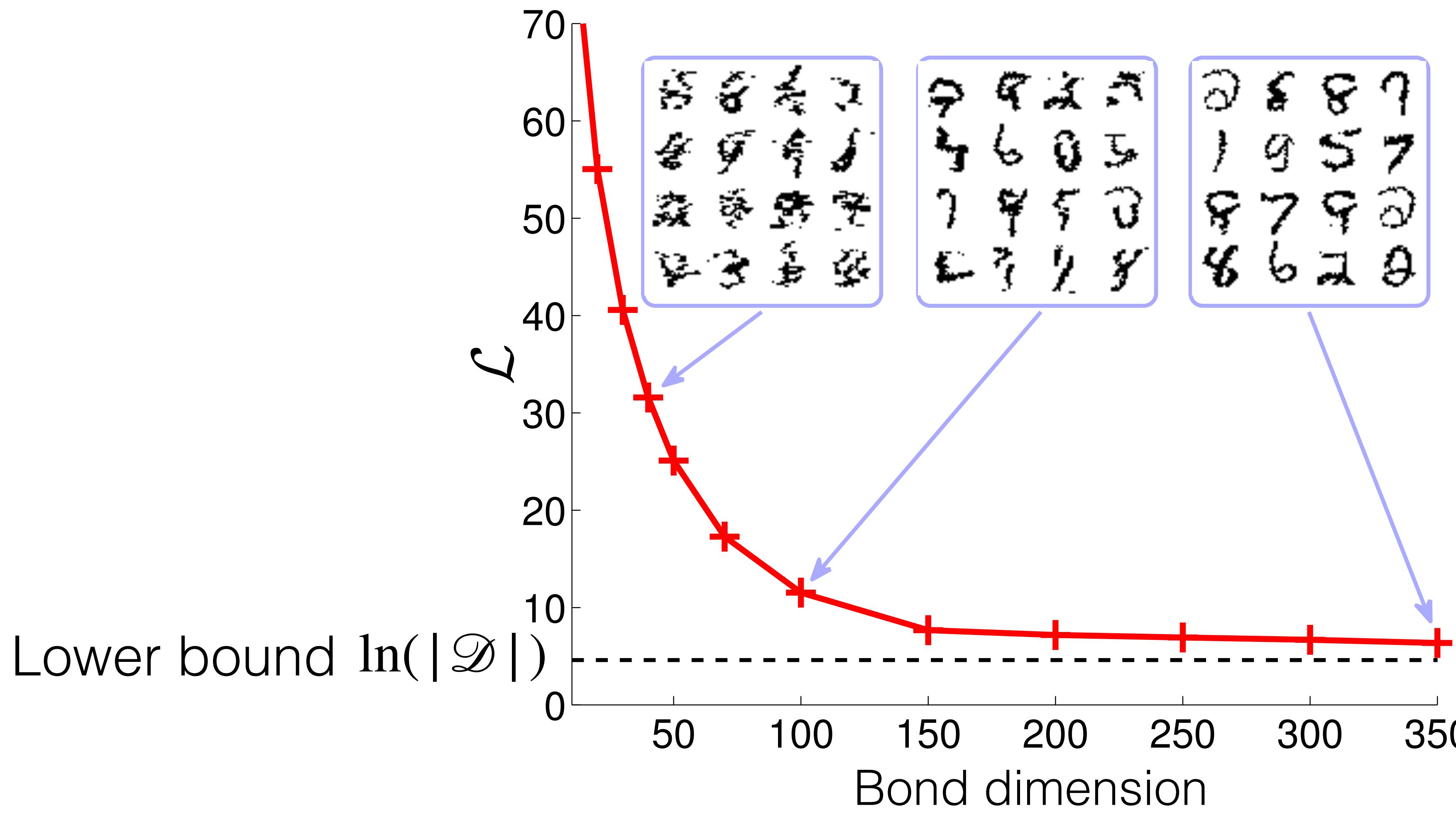
## Tensor Network Machine Learning

Cichocki et al 1604.05271, 1609.00893, 1708.09165...  
Stoudenmire, Schwab NIPS 2016 Liu et al 1710.04833  
Stoudenmire Q. Sci. Tech. 2018 Liu et al 1803.09111

Novikov et al 1509.06569  
Hallam et al 1711.03357  
Glasser et al 1806.05964

Kossaifi et al 1707.08308  
Gallego, Orus 1708.01525  
Pestun et al 1711.01416...

# Name of the game



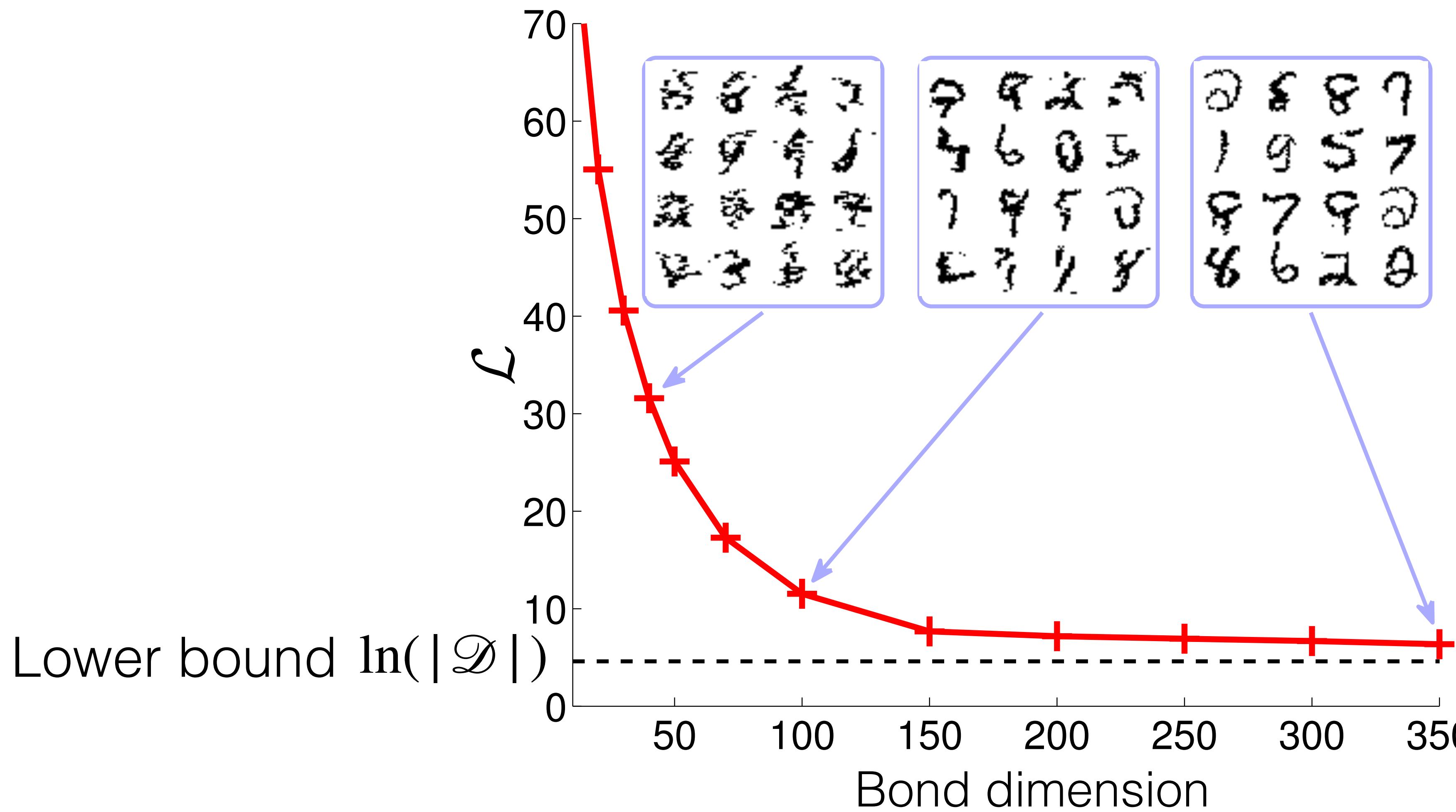
$$\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ln p(x)$$

$\downarrow$

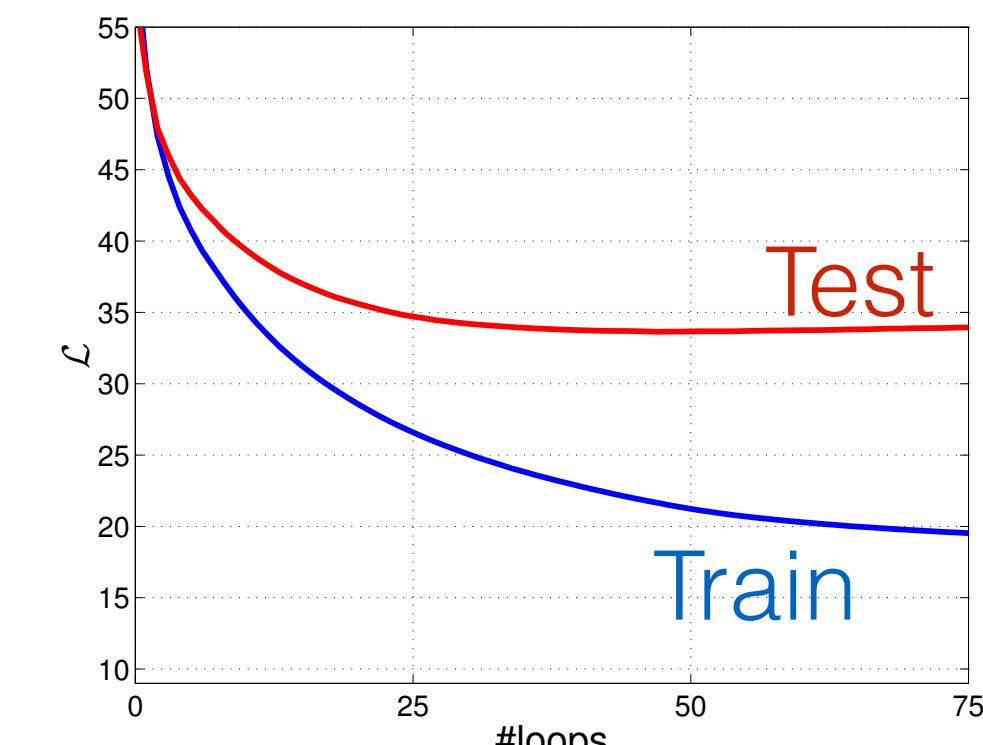
$$|\text{---}|^2 / Z$$

The ultimate goal is “learn to forget”

# Name of the game



The ultimate goal is “learn to forget”

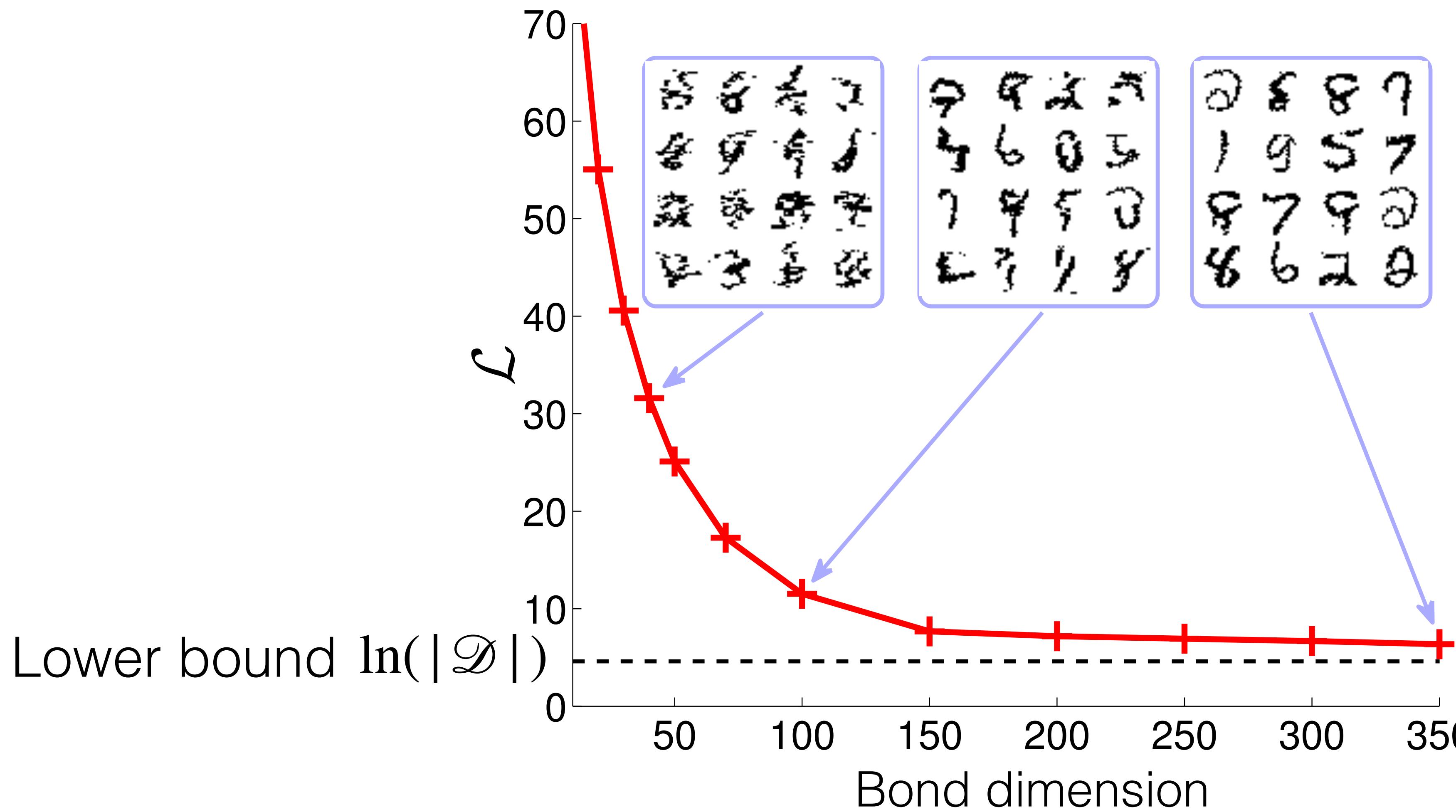


$$\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ln p(x)$$

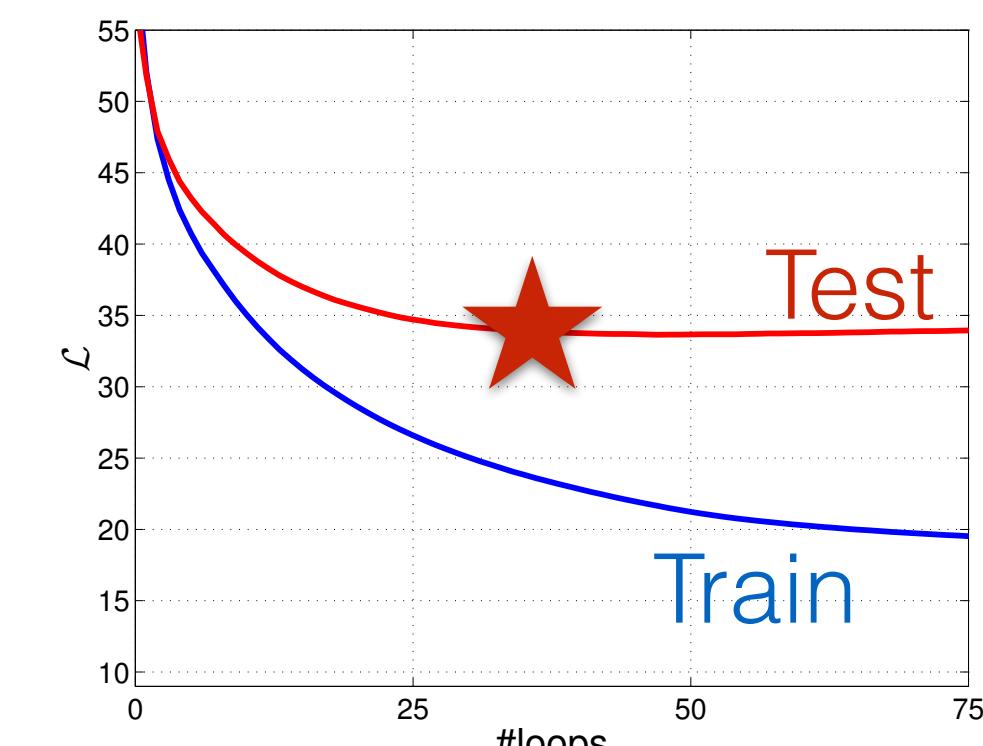
$\downarrow$

$$|\langle \bullet \bullet \bullet \bullet \rangle|^2 / Z$$

# Name of the game



The ultimate goal is “learn to forget”



$$\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ln p(x)$$

$$|\text{---}|^2 / Z$$

# *Nice features of an MPS Born Machine*

$$p(x) = |\text{---|---|---}|^2 / Z$$

**Learning**

**Inference**

**Sampling**

**Expressibility**

Glasser, Clark, Deng,  
Gao, Chen, Huang... 17'

# Tractable Likelihood

$$Z = \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \dots \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \dots \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array}$$

tractable via  
efficient tensor contraction

$$\partial Z / \partial (\begin{array}{c} \text{---} \\ | \\ \text{---} \end{array}) = 2 \times \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \dots \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array}$$

\*applies to TTN  
and MERA as well

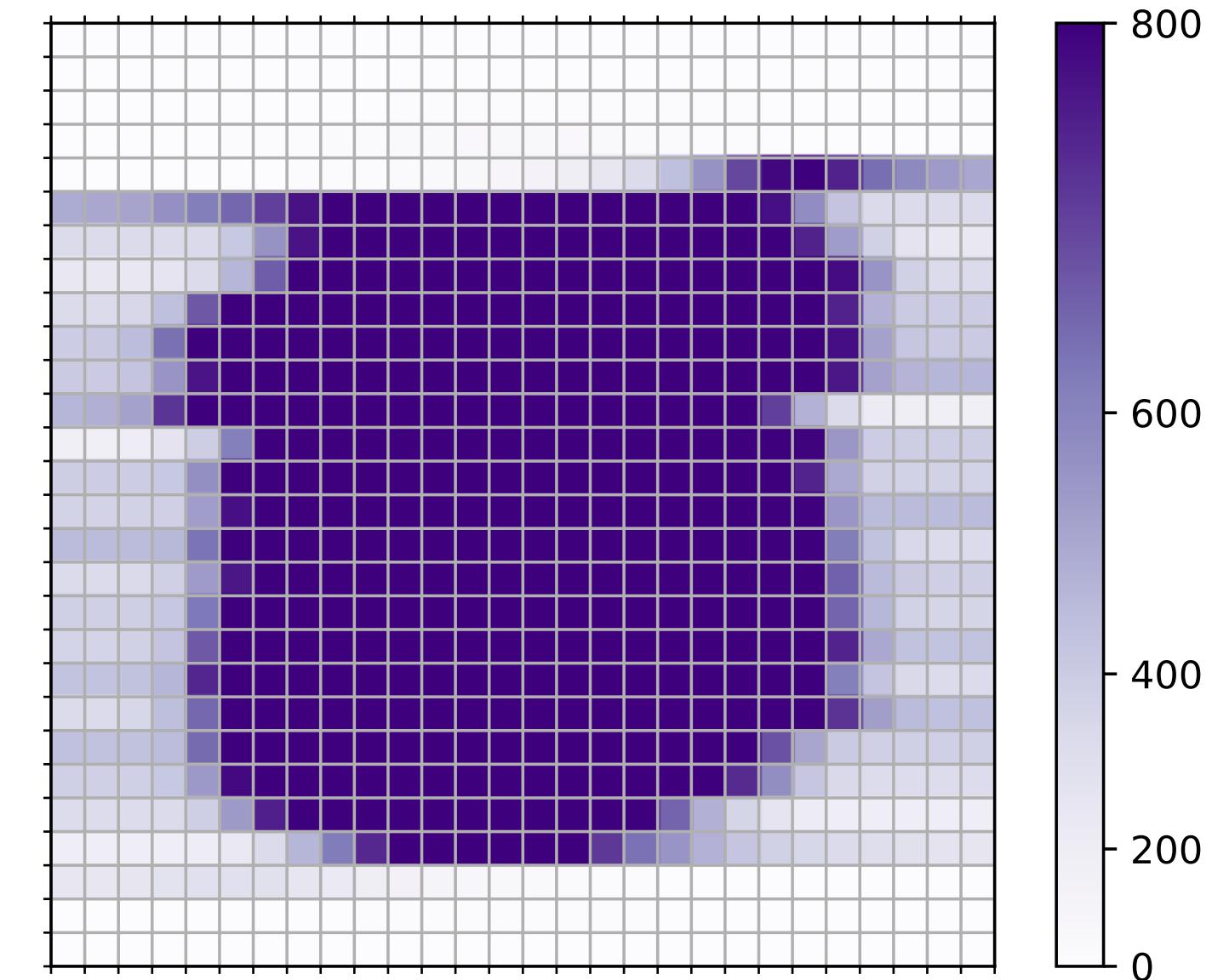
**Efficient & unbiased learning compared to  
models with intractable partition functions**

# Adaptive Learning

Training images



Bond dimensions



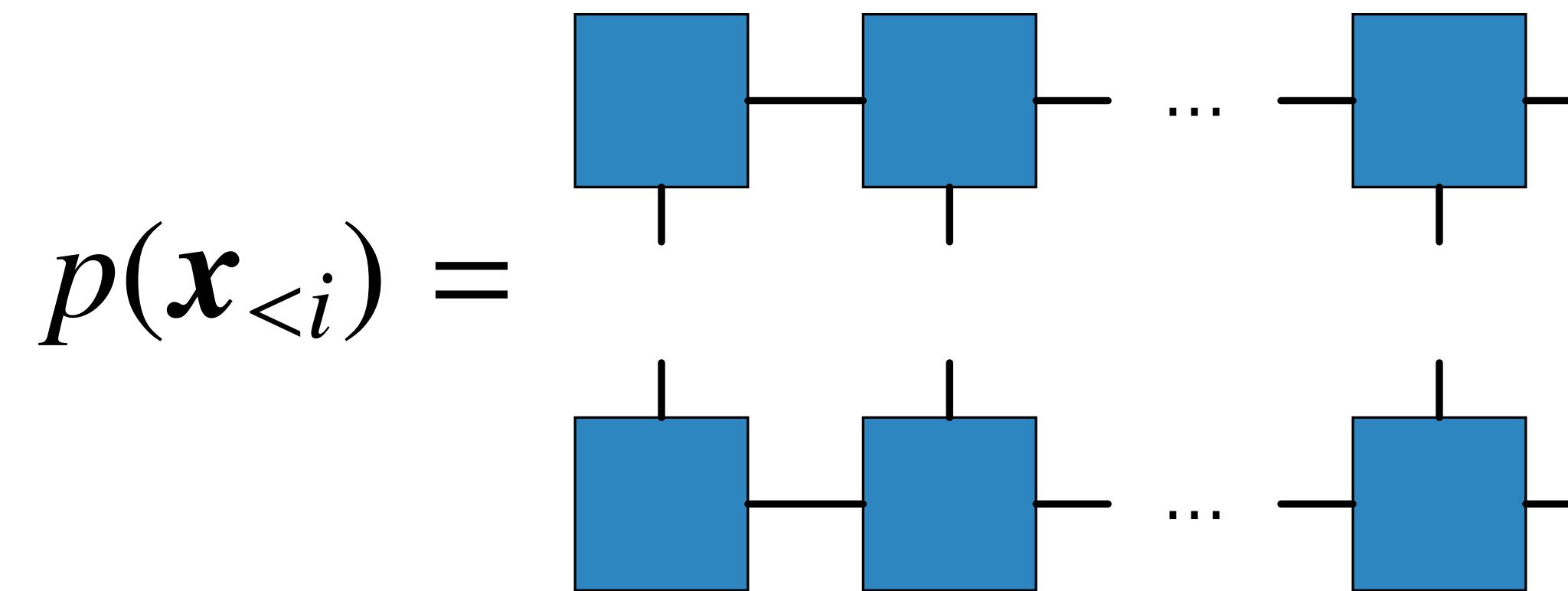
\*applies to 2-site optimization

**Adaptively grows the bond dimensions, thus dynamically tuning the expressibility**

# Direct Generation

$$p(\mathbf{x}) = \prod_i p(x_i | \mathbf{x}_{<i}) = \prod_i \frac{p(\mathbf{x}_{$$

Ferris & Vidal 2012



\*applies to TTN  
and MERA as well

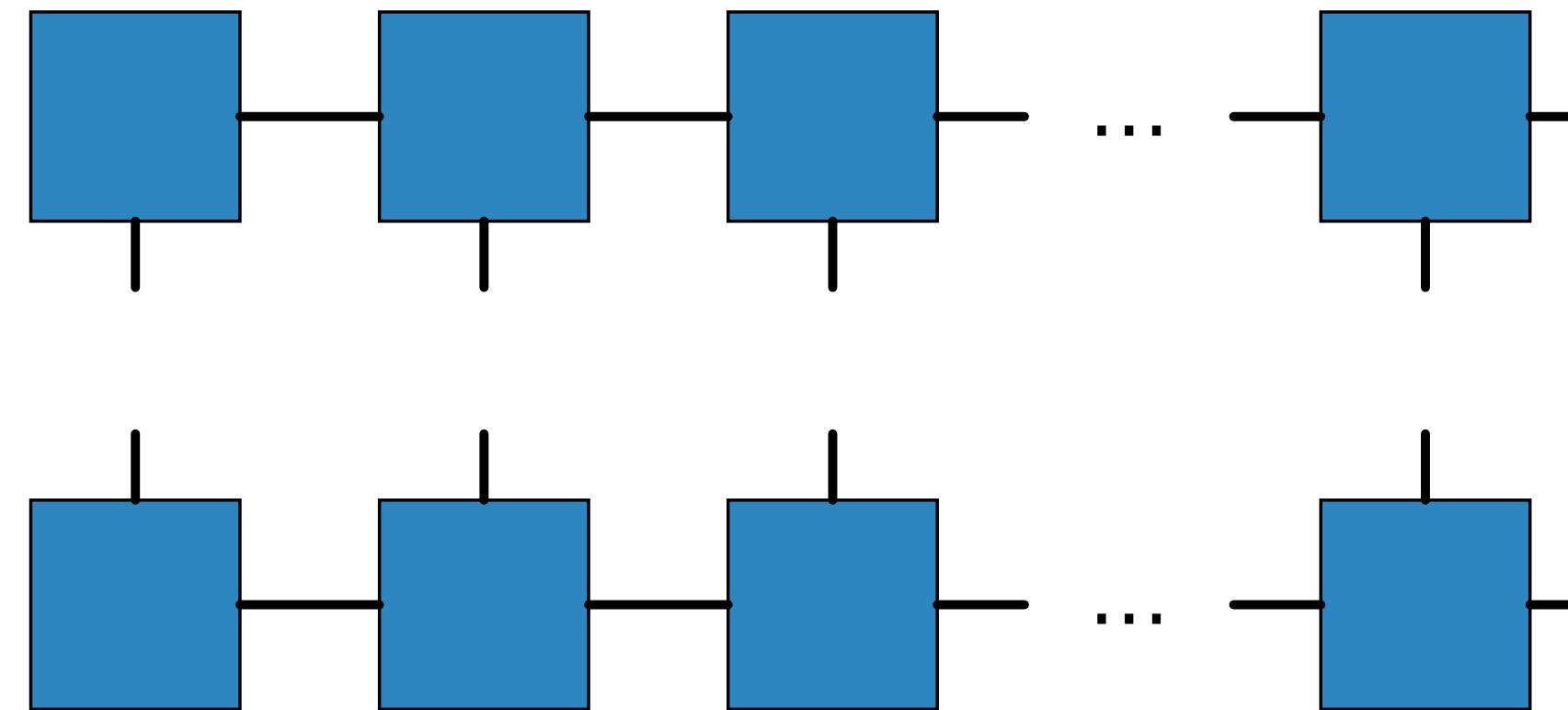
No thermalization issue compared to  
slow mixing Gibbs sampling of Boltzmann Machines

# Direct Generation

$$p(\mathbf{x}) = \prod_i p(x_i | \mathbf{x}_{<i}) = \prod_i \frac{p(\mathbf{x}_{$$

Ferris & Vidal 2012

$$p(\mathbf{x}_{$$



\*applies to TTN  
and MERA as well

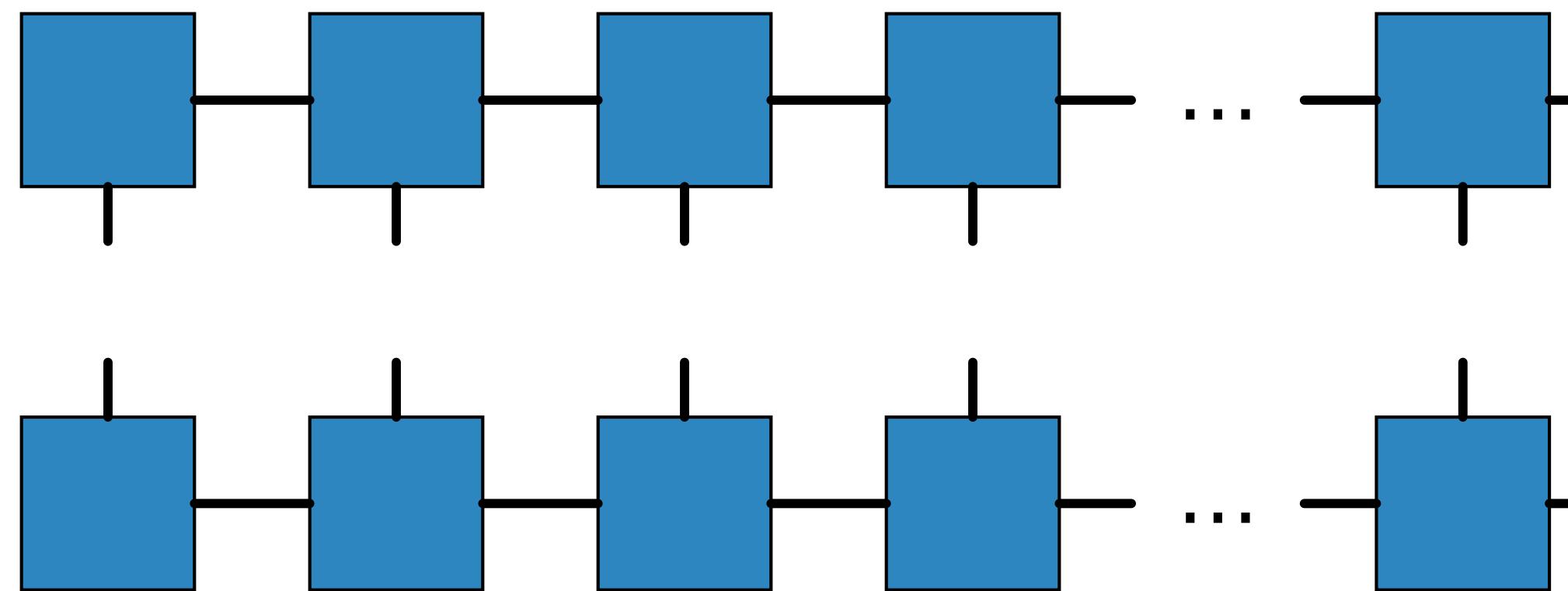
No thermalization issue compared to  
slow mixing Gibbs sampling of Boltzmann Machines

# Direct Generation

$$p(\mathbf{x}) = \prod_i p(x_i | \mathbf{x}_{<i}) = \prod_i \frac{p(\mathbf{x}_{$$

Ferris & Vidal 2012

$$p(\mathbf{x}_{$$



\*applies to TTN  
and MERA as well

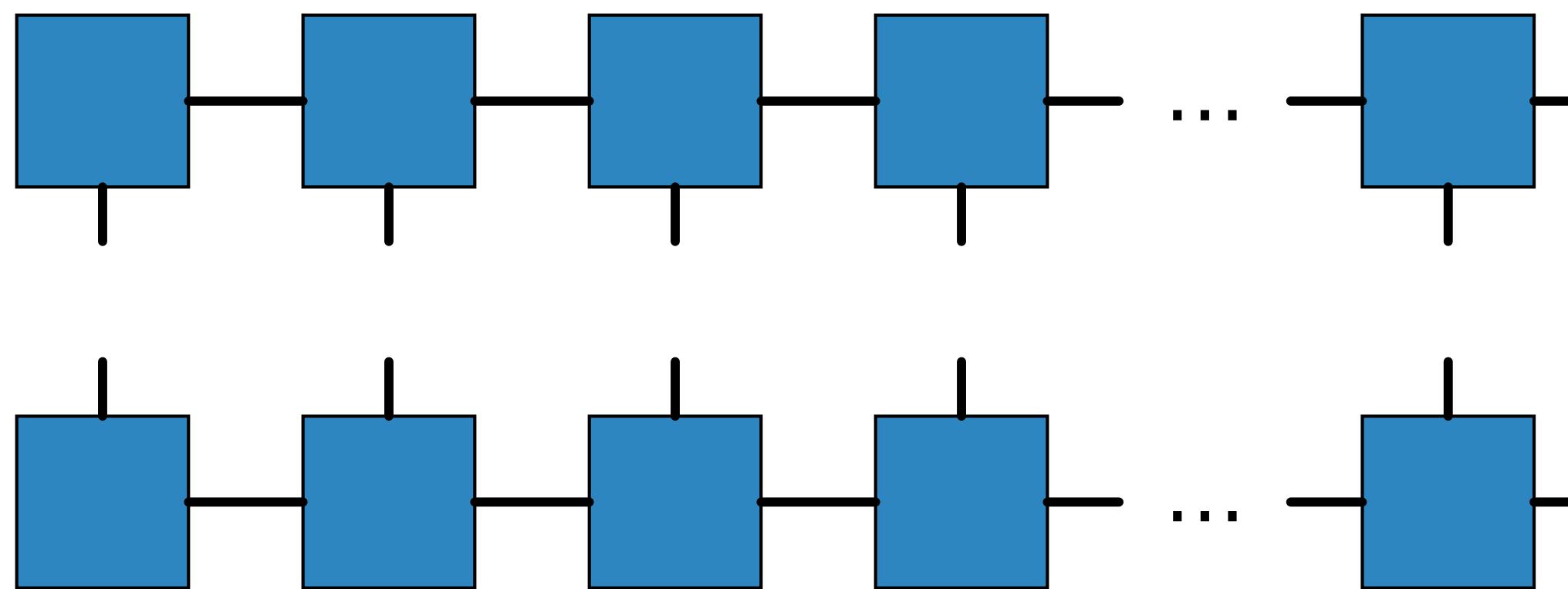
No thermalization issue compared to  
slow mixing Gibbs sampling of Boltzmann Machines

# Direct Generation

$$p(\mathbf{x}) = \prod_i p(x_i | \mathbf{x}_{<i}) = \prod_i \frac{p(\mathbf{x}_{$$

Ferris & Vidal 2012

$$p(\mathbf{x}_{$$



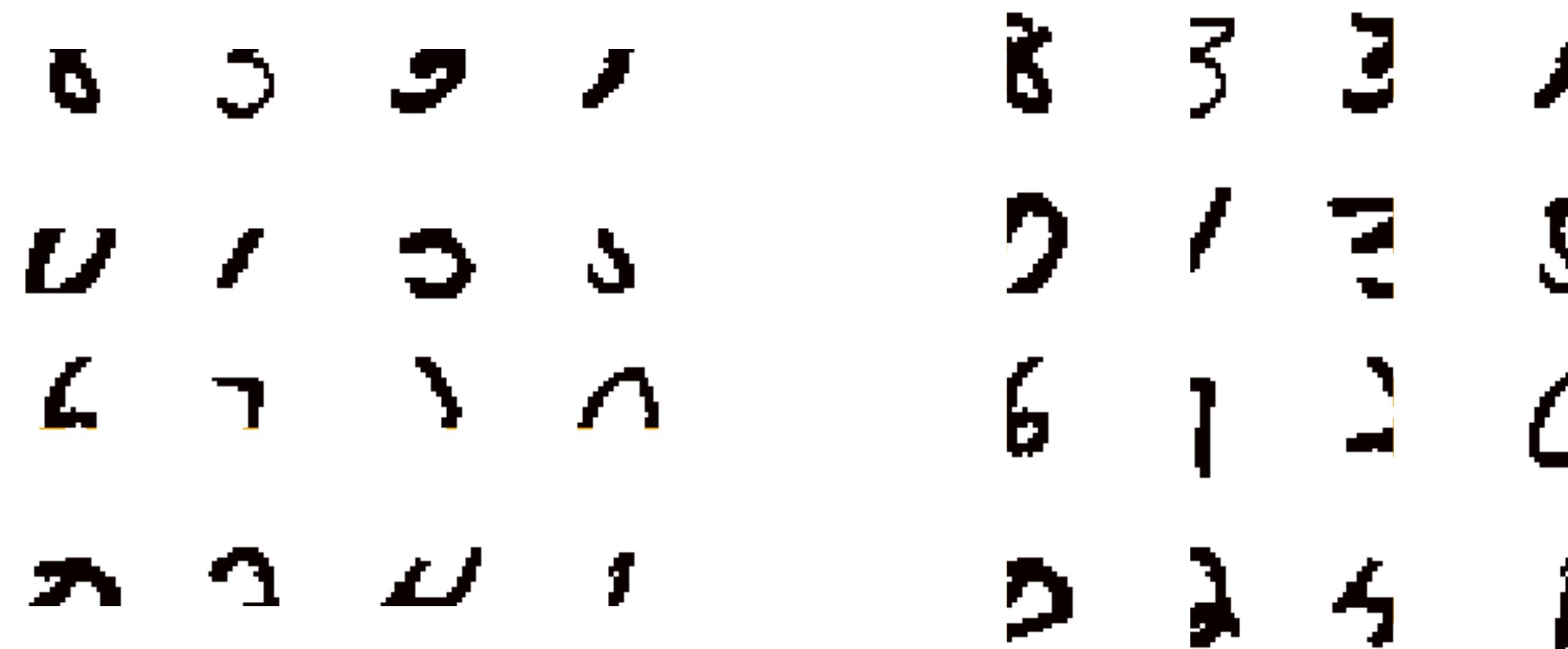
*"Zipper Sampling"*

\*applies to TTN  
and MERA as well

No thermalization issue compared to  
slow mixing Gibbs sampling of Boltzmann Machines

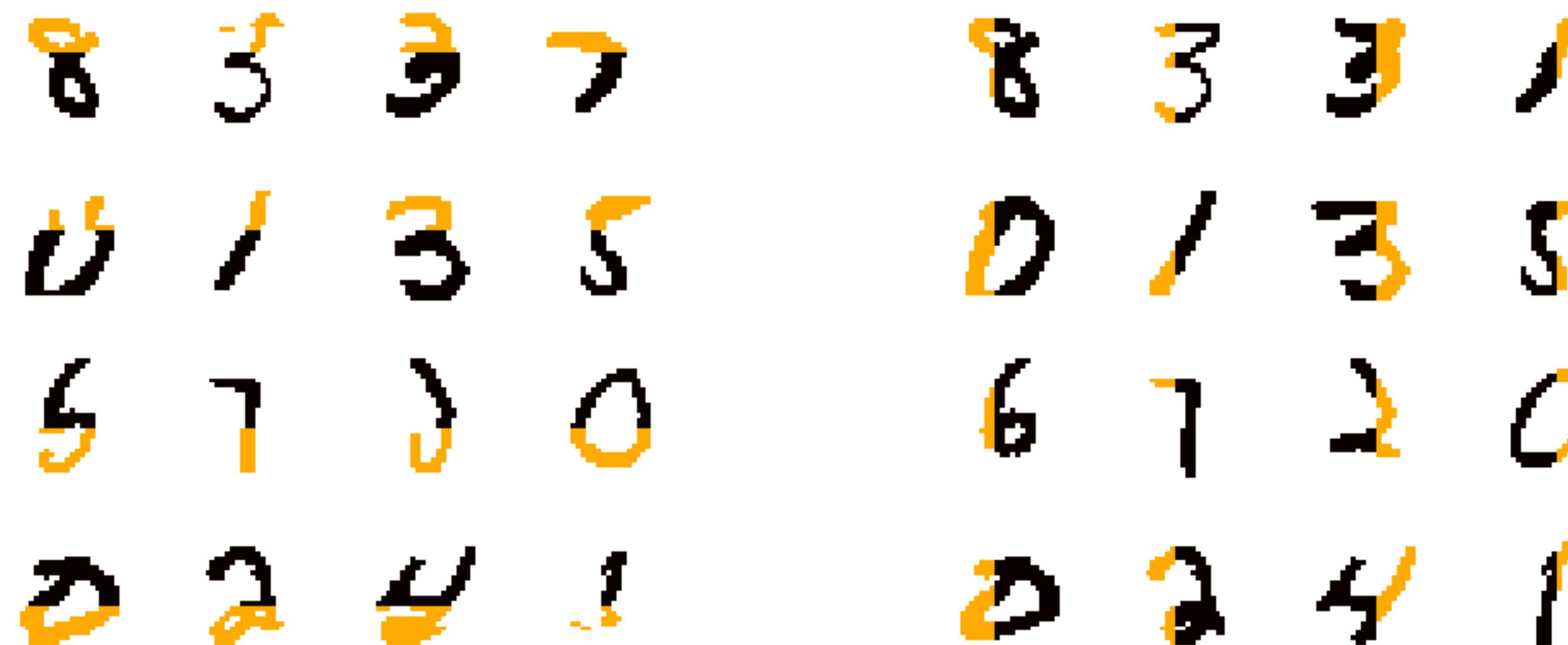
# Image Restoration

Han, Wang, Fan, LW, Zhang, 1709.01662, PRX 18'



# Image Restoration

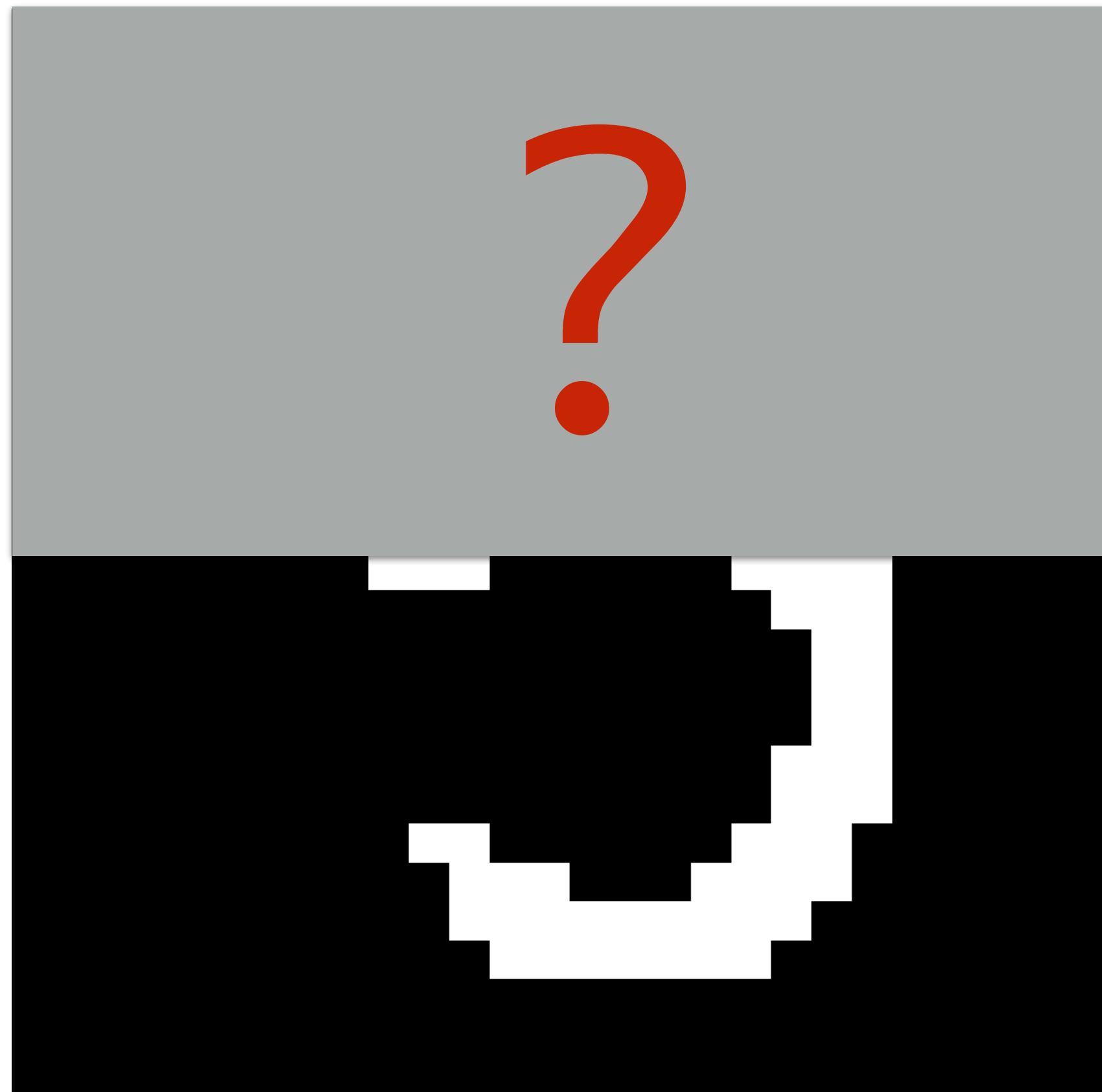
Han, Wang, Fan, LW, Zhang, 1709.01662, PRX 18'



Arbitrary order, in contrast to autoregressive models



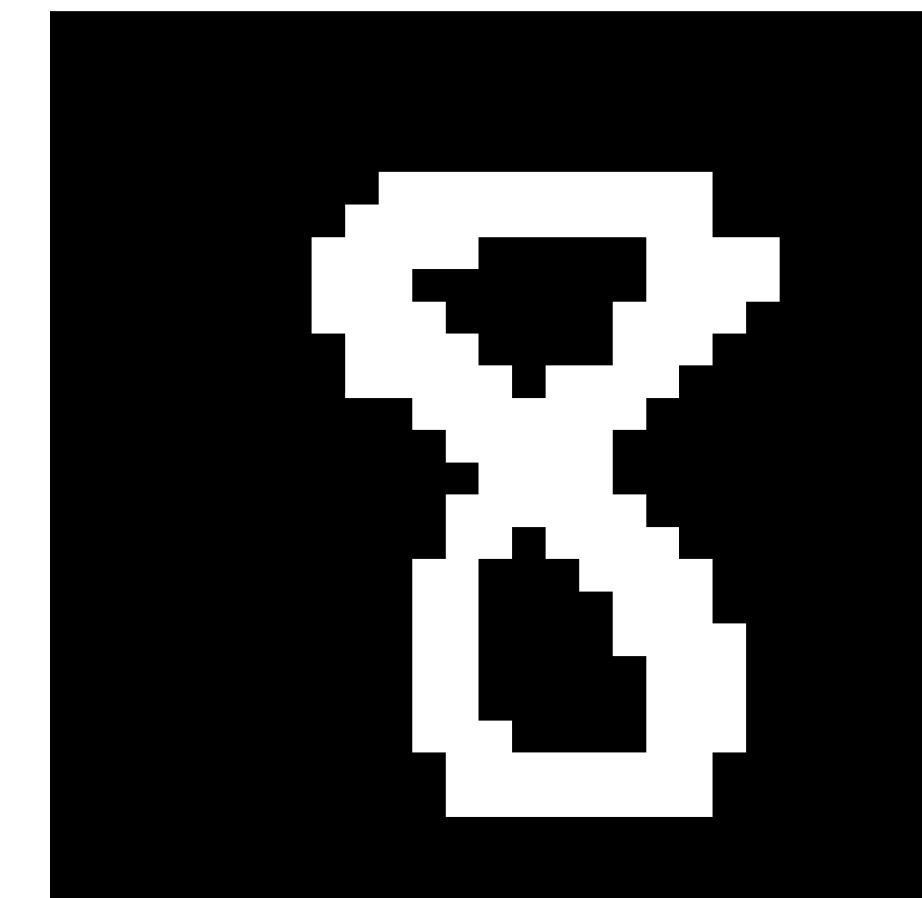
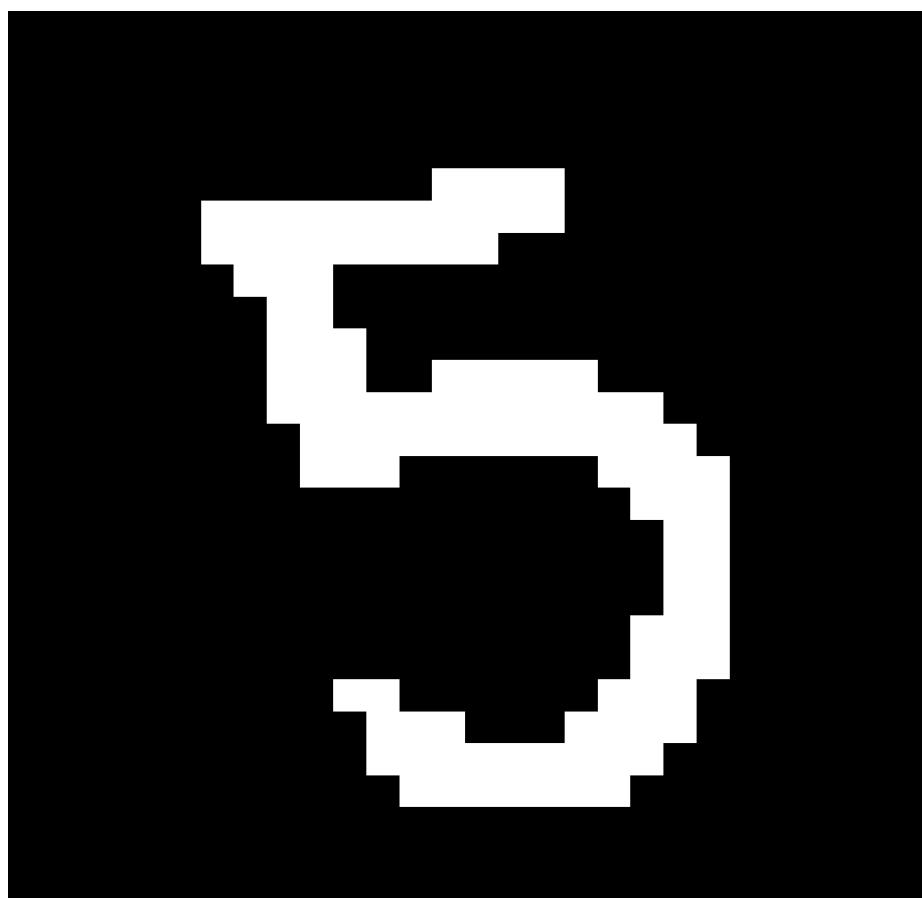
# Quantum Perspective on Deep Learning



# Quantum Perspective on Deep Learning

**Q: How to quantify our inductive biases ?**

**A: Information pattern of probability distributions**



# Quantum Perspective on Deep Learning

**Q: How to quantify our inductive biases ?**

**A: Information pattern of probability distributions**



# Quantum Perspective on Deep Learning

**Q: How to quantify our inductive biases ?**

**A: Information pattern of probability distributions**



# Quantum Perspective on Deep Learning

$$p\left(\begin{array}{|c|} \hline \text{E} \\ \hline \text{3} \\ \hline \end{array}\right) \times p\left(\begin{array}{|c|} \hline \text{3} \\ \hline \text{E} \\ \hline \end{array}\right)$$

---

$$p\left(\begin{array}{|c|} \hline \text{E} \\ \hline \text{3} \\ \hline \end{array}\right) \times p\left(\begin{array}{|c|} \hline \text{3} \\ \hline \text{E} \\ \hline \end{array}\right)$$

# Quantum Perspective on Deep Learning

## Classical mutual information

$$I = - \left\langle \ln \left\langle \frac{p(x, y') p(x', y)}{p(x', y') p(x, y)} \right\rangle_{x', y'} \right\rangle_{x, y}$$

## Quantum Renyi entanglement entropy

$$S = - \ln \left\langle \left\langle \frac{\Psi(x, y') \Psi(x', y)}{\Psi(x', y') \Psi(x, y)} \right\rangle_{x', y'} \right\rangle_{x, y}$$

**Striking similarity implies common inductive bias**

- + Quantitative & interpretable approaches
- + Principled structure design & learning

Cheng, Chen, LW,  
1712.04144, Entropy 18'

# DEEP LEARNING AND QUANTUM ENTANGLEMENT: FUNDAMENTAL CONNECTIONS WITH IMPLICATIONS TO NETWORK DESIGN

**Yoav Levine, David Yakira, Nadav Cohen & Amnon Shashua**

The Hebrew University of Jerusalem

{yoavlevine, davidyakira, cohennadav, shashua}@cs.huji.ac.il

## ABSTRACT

Formal understanding of the inductive bias behind deep convolutional networks, i.e. the relation between the network’s architectural features and the functions it is able to model, is limited. In this work, we establish a fundamental connection between the fields of quantum physics and deep learning, and use it for obtaining novel theoretical observations regarding the inductive bias of convolutional networks. Specifically, we show a structural equivalence between the function realized by a convolutional arithmetic circuit (ConvAC) and a quantum many-body wave function, which facilitates the use of quantum entanglement measures as quantifiers of a deep network’s expressive ability to model correlations. Furthermore, the construction of a deep ConvAC in terms of a quantum Tensor Network is enabled. This allows us to perform a graph-theoretic analysis of a convolutional network, tying its expressiveness to a min-cut in its underlying graph. We demonstrate a practical outcome in the form of a direct control over the inductive bias via the number of channels (width) of each layer. We empirically validate our findings on standard convolutional networks which involve ReLU activations and max pooling. The description of a deep convolutional network in well-defined graph-theoretic tools and the structural connection to quantum entanglement, are two interdisciplinary bridges that are brought forth by this work.

# DEEP LEARNING AND QUANTUM ENTANGLEMENT: FUNDAMENTAL CONNECTIONS WITH IMPLICATIONS TO NETWORK DESIGN

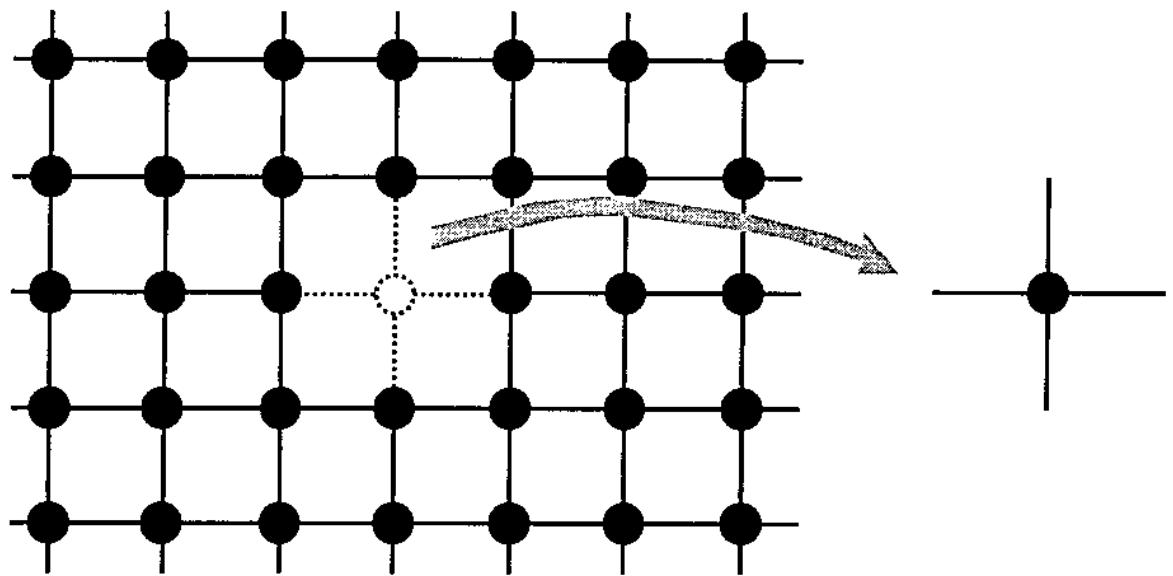
**Yoav Levine, David Yakira, Nadav Cohen & Amnon Shashua**

The Hebrew University of Jerusalem

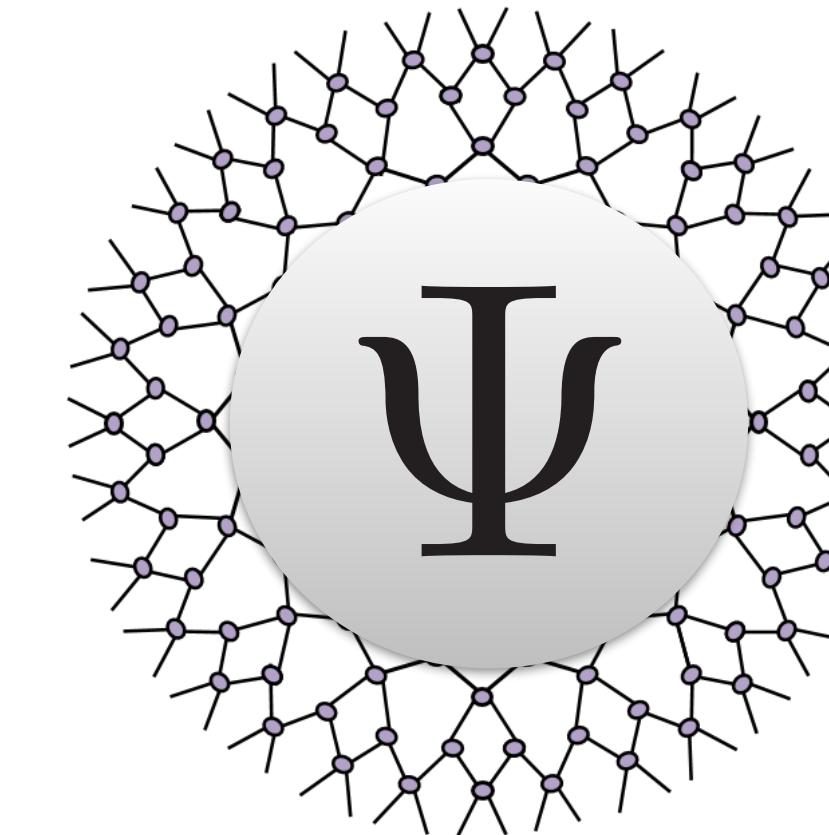


# Physicists' gifts to Machine Learning

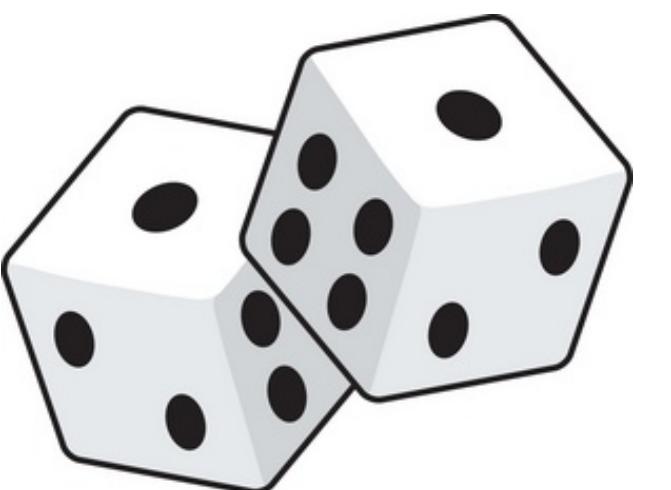
## Mean Field Theory



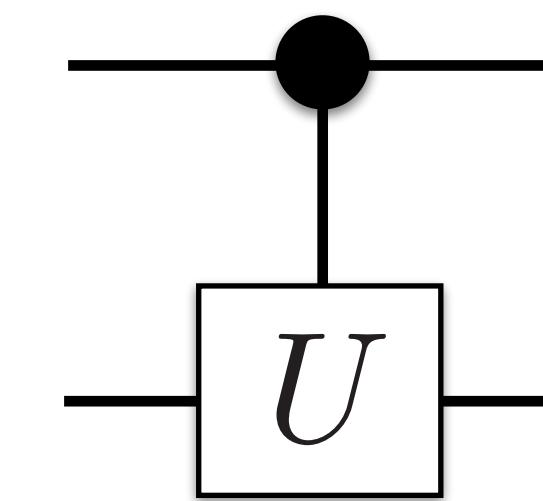
## Tensor Networks



## Monte Carlo Methods

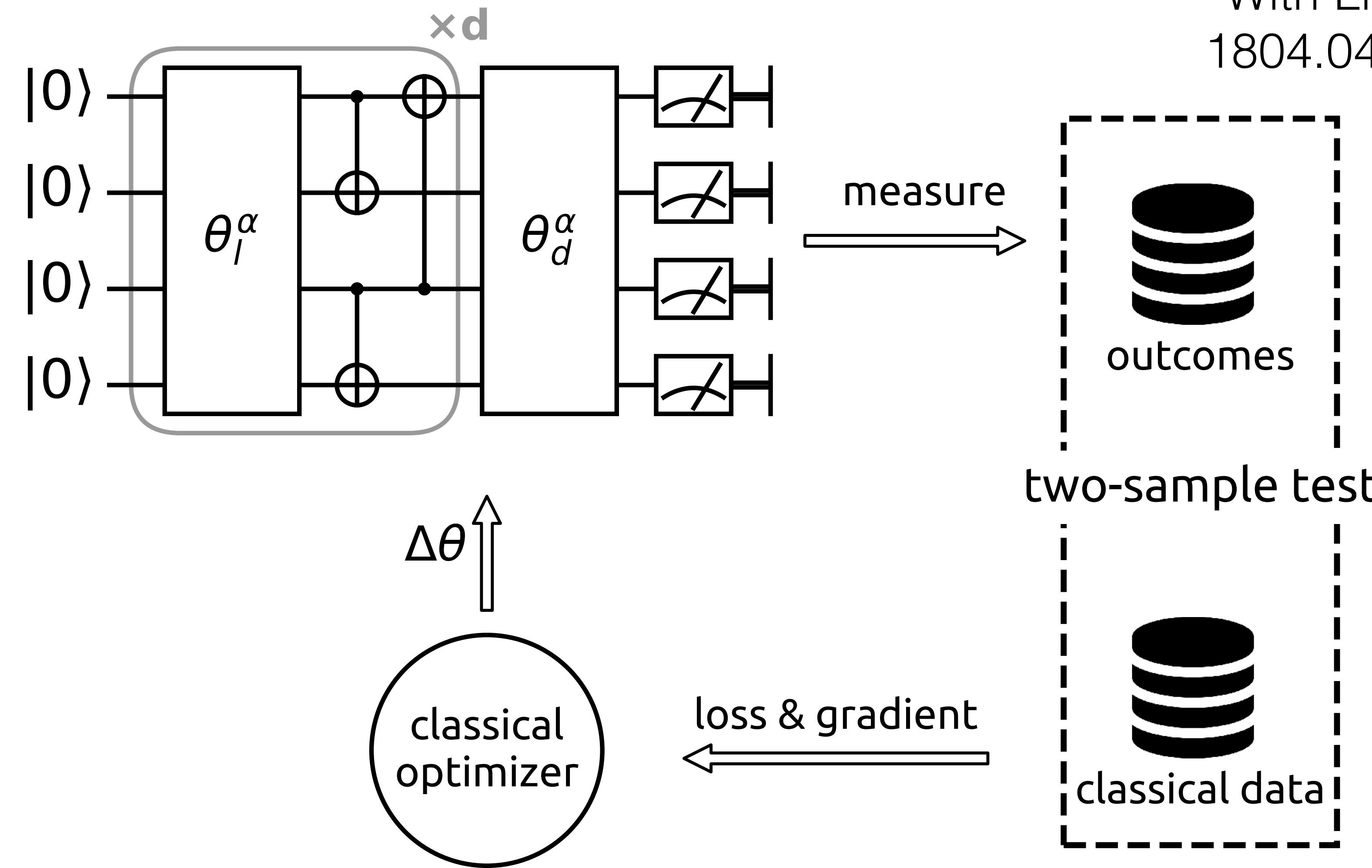


## Quantum Computing



# Quantum Circuit Born Machine

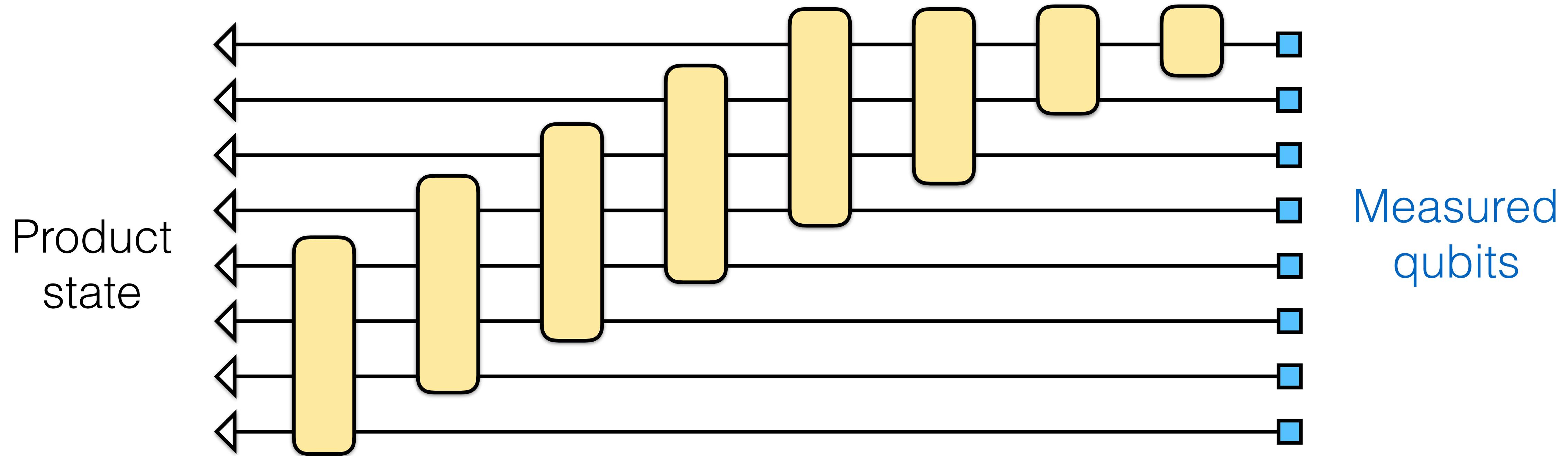
With Liu, Zeng, Wu, Hu  
1804.04168, 1808.03425



Train the quantum circuit as a probabilistic generative model

Quantum sampling complexity underlines the “quantum supremacy”

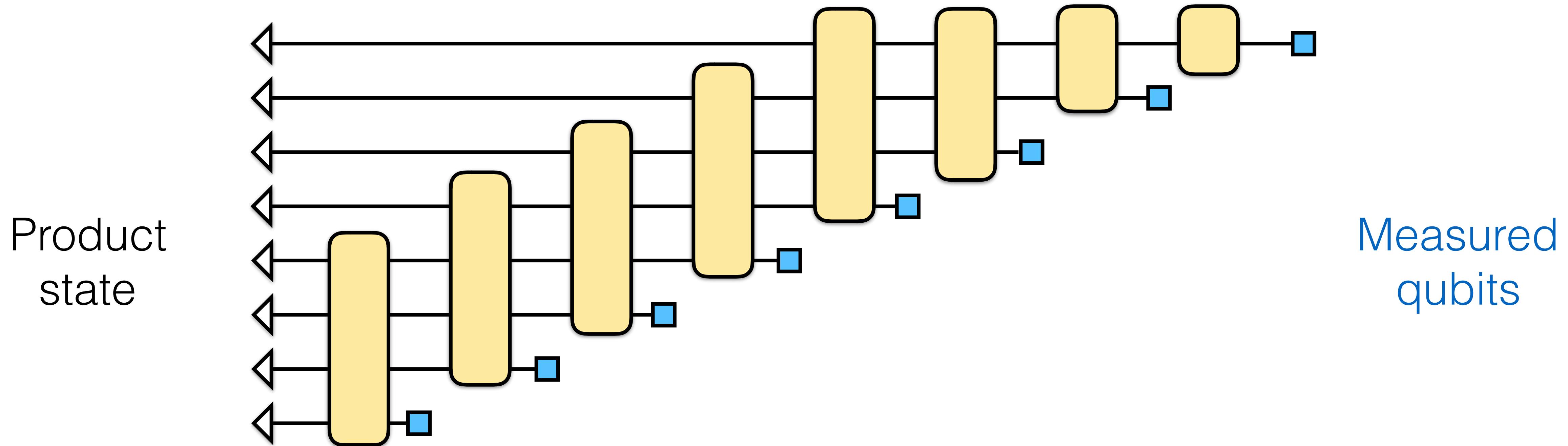
# Qubit efficient scheme



Huggins, Patel, Whaley, Stoudenmire, 1803.11537  
see also Cramer et al, Nat. Comm. 2010

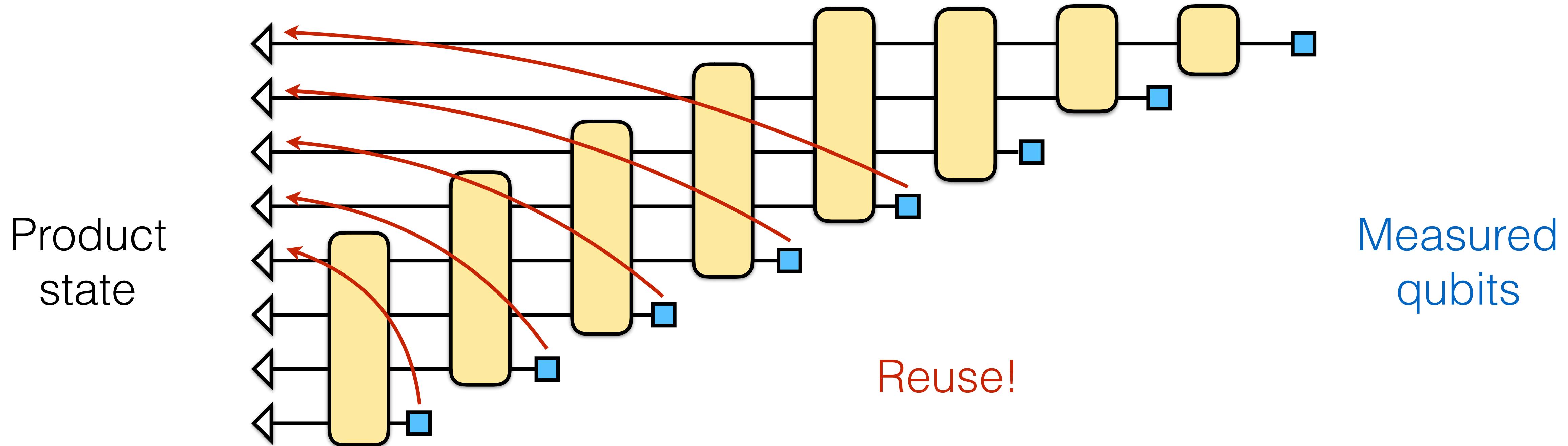
**Tensor network inspired quantum circuit architecture**

# Qubit efficient scheme



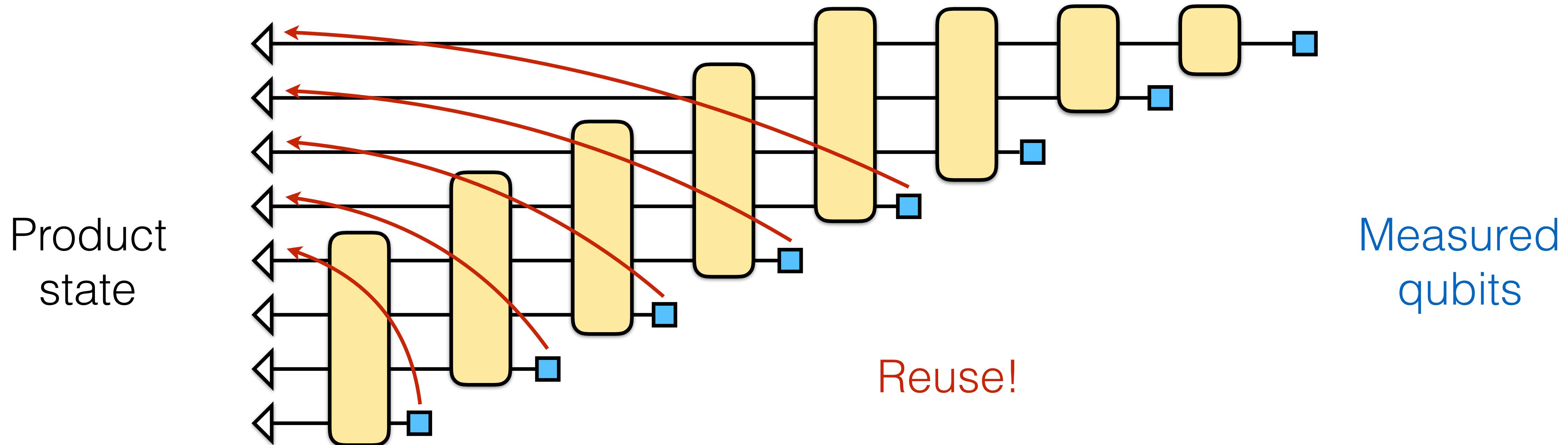
Huggins, Patel, Whaley, Stoudenmire, 1803.11537  
see also Cramer et al, Nat. Comm. 2010

# Qubit efficient scheme

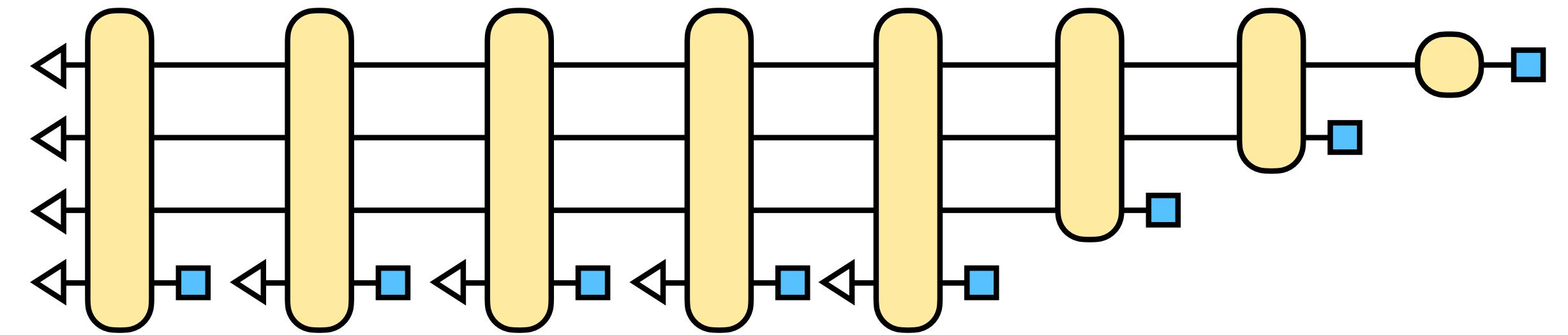


Huggins, Patel, Whaley, Stoudenmire, 1803.11537  
see also Cramer et al, Nat. Comm. 2010

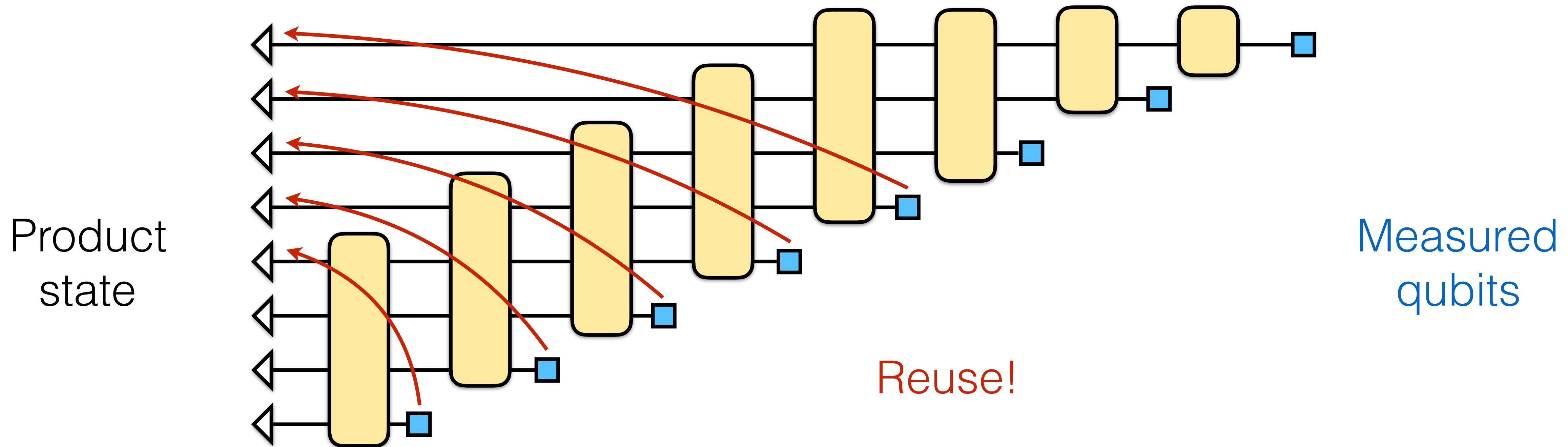
# Qubit efficient scheme



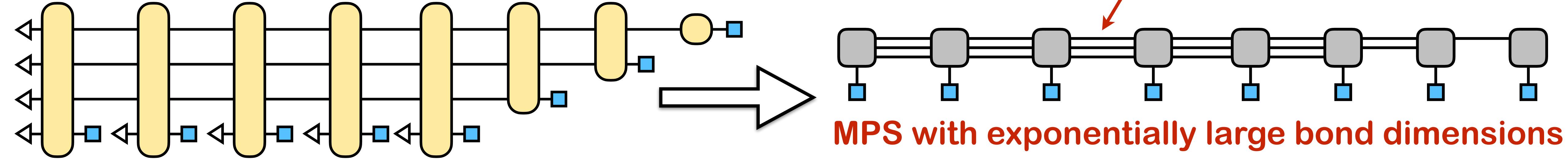
Huggins, Patel, Whaley, Stoudenmire, 1803.11537  
see also Cramer et al, Nat. Comm. 2010



# Qubit efficient scheme



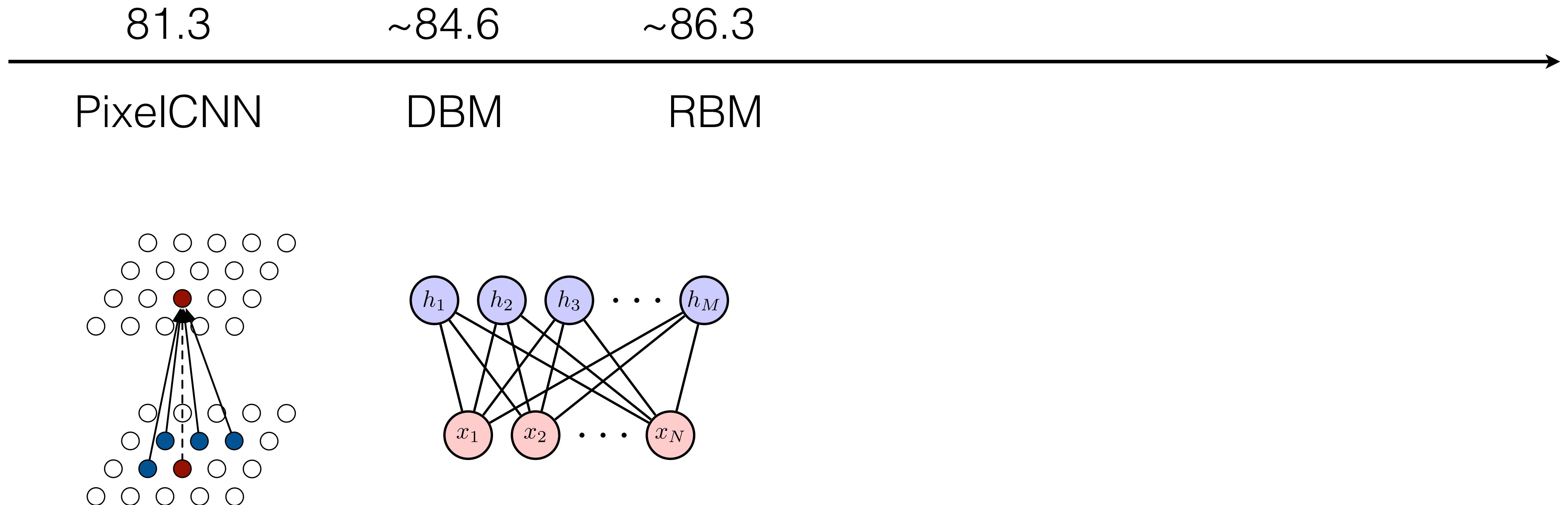
↓  
Huggins, Patel, Whaley, Stoudenmire, 1803.11537  
see also Cramer et al, Nat. Comm. 2010



*Well, back to Boltzmann Machines*

# Quantitative Performance

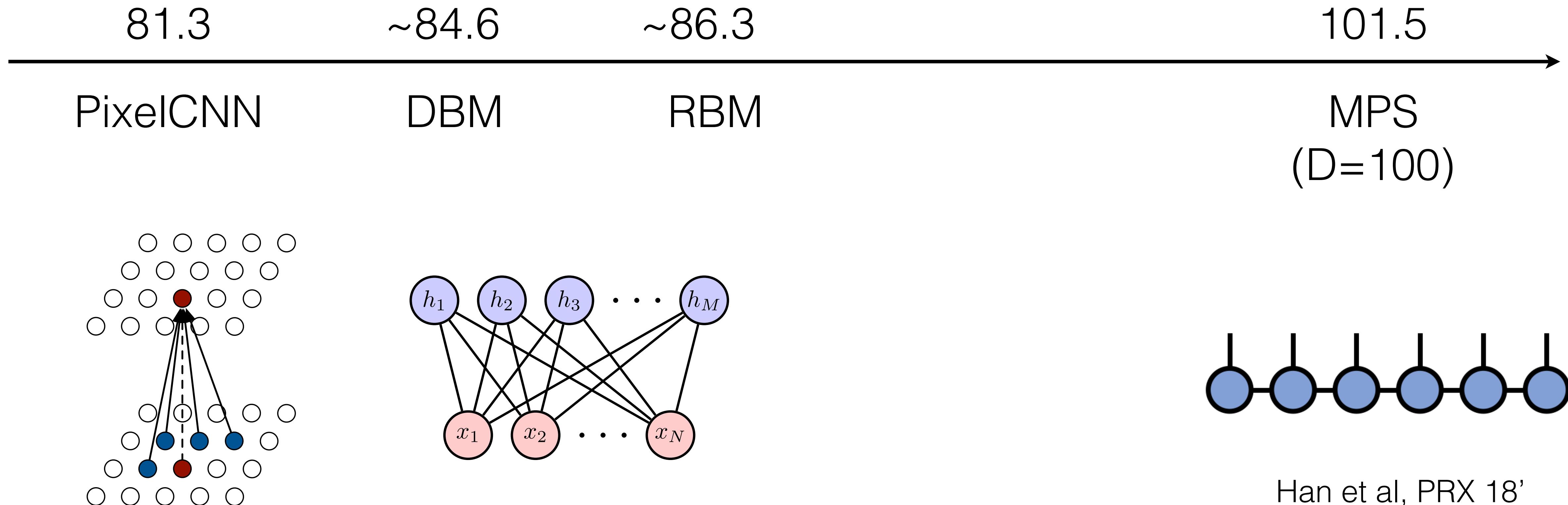
**Binarized MNIST test NLL, lower is better\***



\*Lower test NLL may not imply better sample quality, Theis et al, 1511.01844

# Quantitative Performance

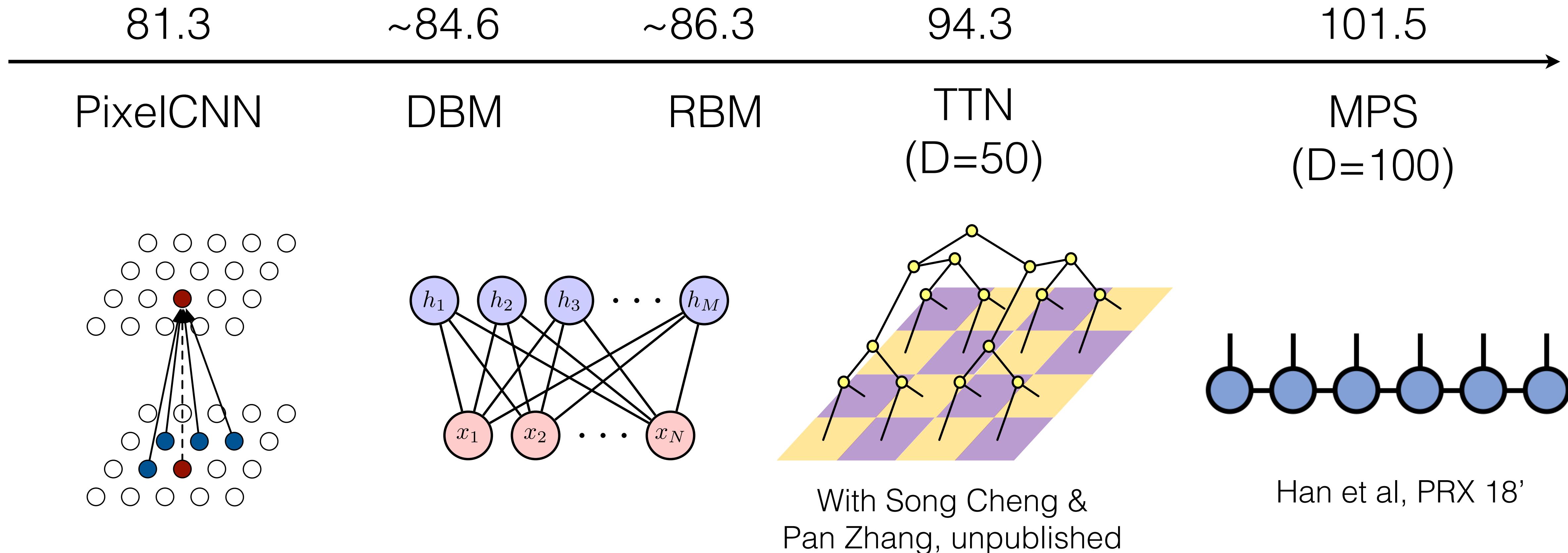
**Binarized MNIST test NLL, lower is better\***



\*Lower test NLL may not imply better sample quality, Theis et al, 1511.01844

# Quantitative Performance

**Binarized MNIST test NLL, lower is better\***



\*Lower test NLL may not imply better sample quality, Theis et al, 1511.01844

# “Positive” PEPS

**Probability  
model**

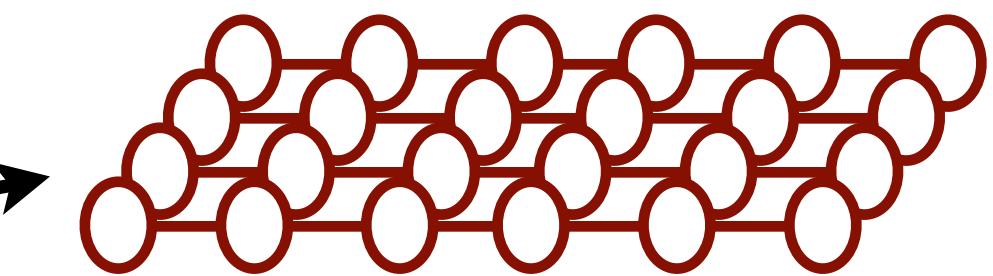
$$p(\mathbf{x}) = \text{PEPS} / Z$$

Why positive?

- Smaller truncation error
- No need for double layer tensor

aka hidden Markov  
random field model

$$p(\mathbf{x}) = \sum_{\mathbf{h}} \prod_i f^i(x_i, \mathbf{h}) / Z$$



$$\text{---} \circ \text{---} = \sum_{\sigma} \text{---} \circ \text{---}^{\sigma}$$

# “Positive” PEPS

aka hidden Markov  
random field model

$$p(\mathbf{x}) = \sum_{\mathbf{h}} \prod_i f^i(x_i, \mathbf{h}) / Z$$

## Probability model

$$p(\mathbf{x}) = \text{Diagram} / Z \rightarrow \text{Diagram}$$

Why positive?

- Smaller truncation error
- No need for double layer tensor

$$\text{Diagram} = \sum_{\sigma} \text{Diagram}$$

## Loss function

$$\mathcal{L} = - \langle \ln (\text{Diagram}) \rangle_{\mathbf{x} \sim \mathcal{D}} + \ln (\text{Diagram})$$

Contract for each given sample

Contract once

# “Positive” PEPS

aka hidden Markov  
random field model

$$p(x) = \sum_h \prod_i f^i(x_i, h) / Z$$

## Probability model

$$p(x) = \text{Diagram} / Z \rightarrow \text{Diagram}$$

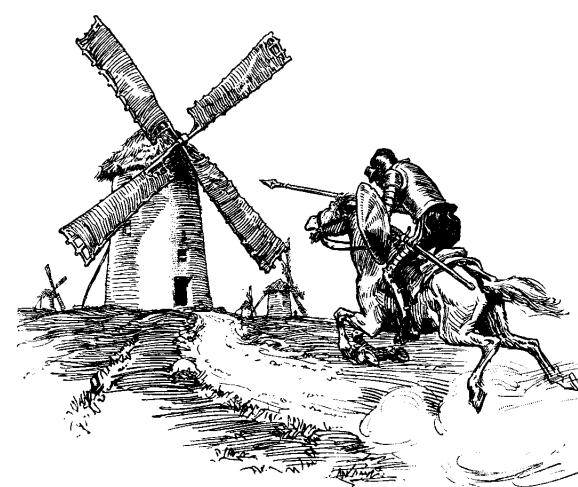
Why positive?

- Smaller truncation error
- No need for double layer tensor

$$\text{Diagram} = \sum_{\sigma} \text{Diagram}$$

## Loss function

$$\mathcal{L} = - \langle \ln (\text{Diagram}) \rangle_{x \sim \mathcal{D}} + \ln (\text{Diagram})$$



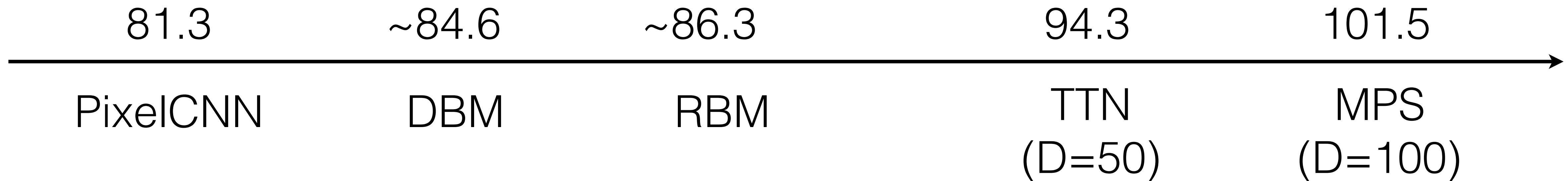
Contract for each given sample

Contract once

**Contract 28x28 PEPS 60000+1 times per epoch!**

# Quantitative Performance

**Binarized MNIST test NLL, lower is better**



**Loss function**

$$\mathcal{L} = - \langle \ln \left( \text{blue network} \right) \rangle_{x \sim \mathcal{D}} + \ln \left( \text{red network} \right)$$



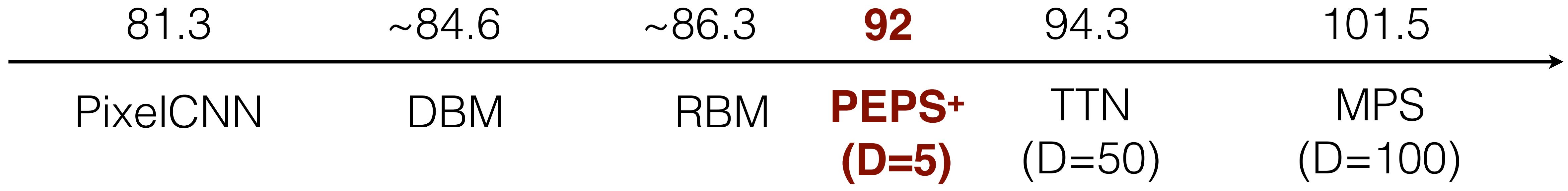
Contract for each given sample

Contract once

**Contract 28x28 PEPS 60000+1 times per epoch!**

# Quantitative Performance

**Binarized MNIST test NLL, lower is better**



With Song Cheng &  
Pan Zhang, unpublished

## Loss function

$$\mathcal{L} = - \left\langle \ln \left( \text{blue network diagram} \right) \right\rangle_{x \sim \mathcal{D}} + \ln \left( \text{red network diagram} \right)$$



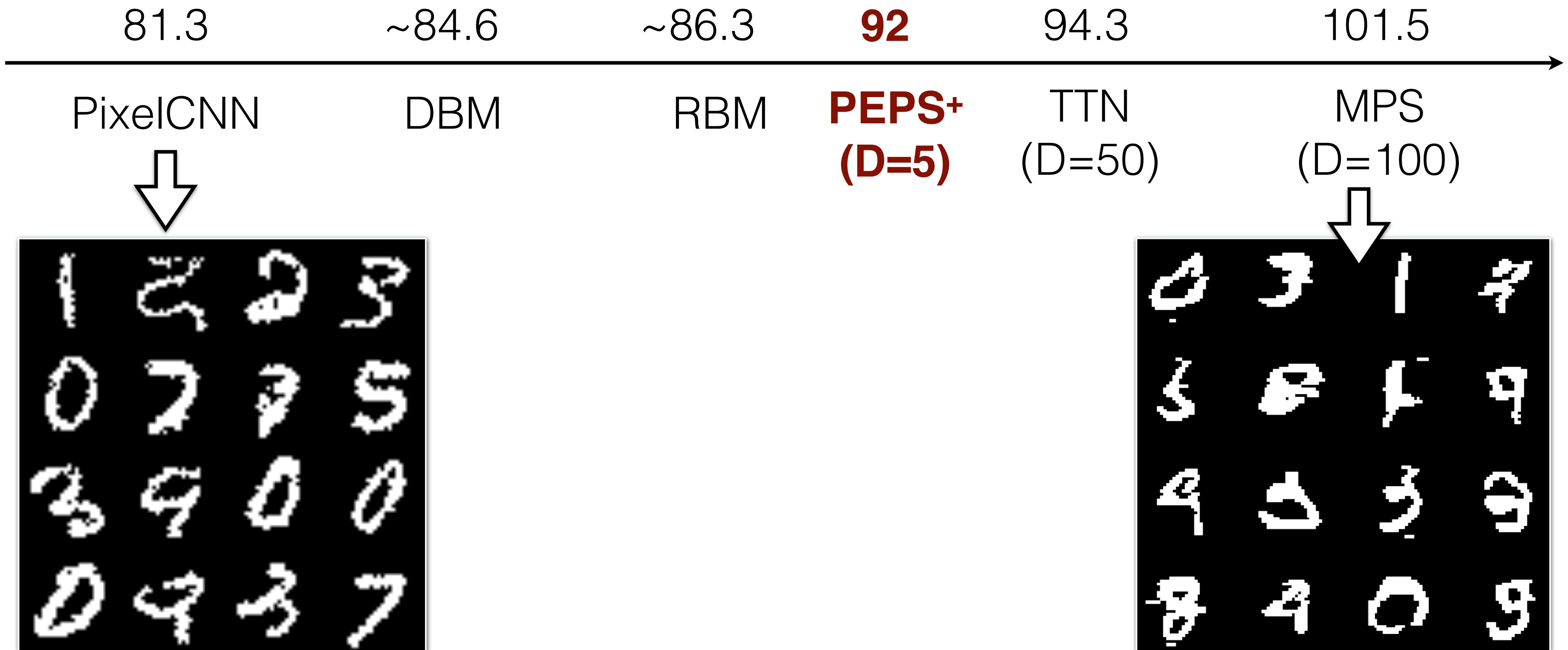
Contract for each given sample

Contract once

**Contract 28x28 PEPS 60000+1 times per epoch!**

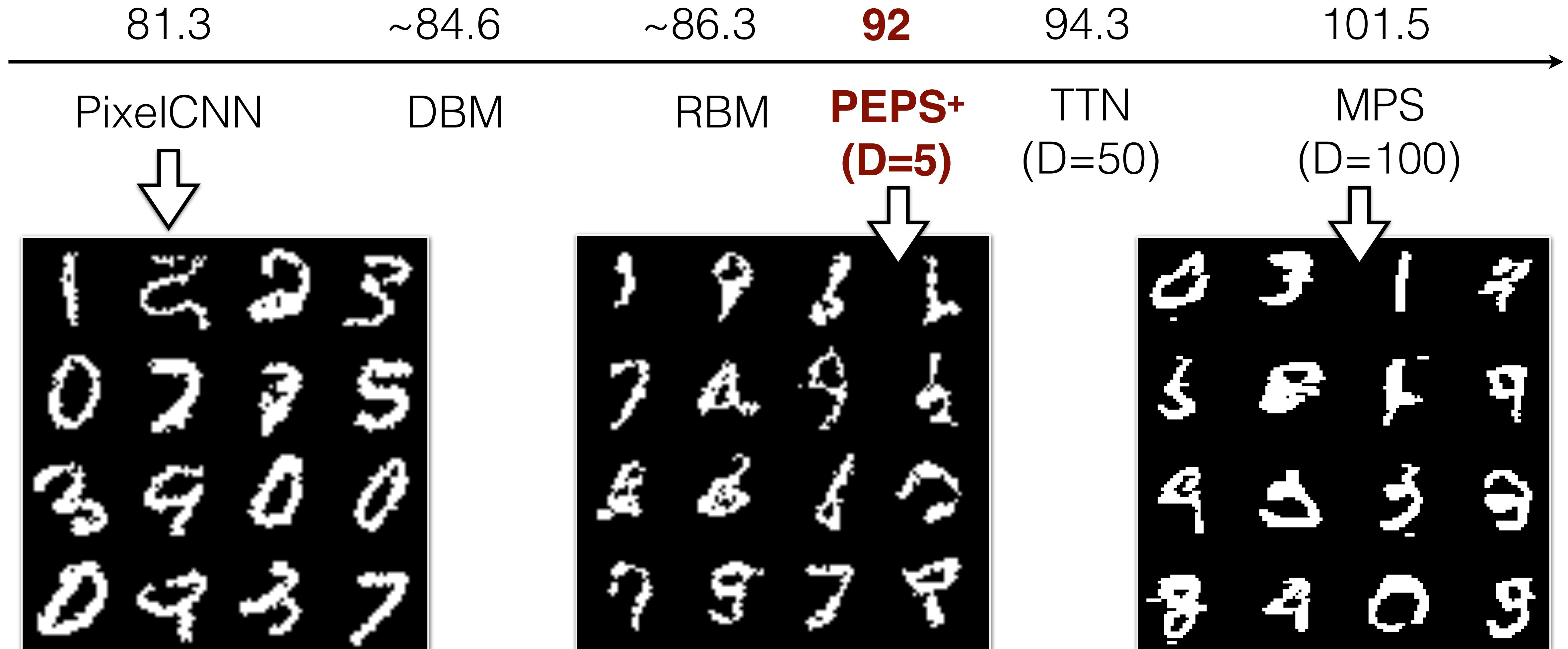
# Quantitative Performance

**Binarized MNIST test NLL, lower is better**



# Quantitative Performance

**Binarized MNIST test NLL, lower is better**



*Bonus:*  
*What does deep learning offer to  
tensor network states & algorithms?*

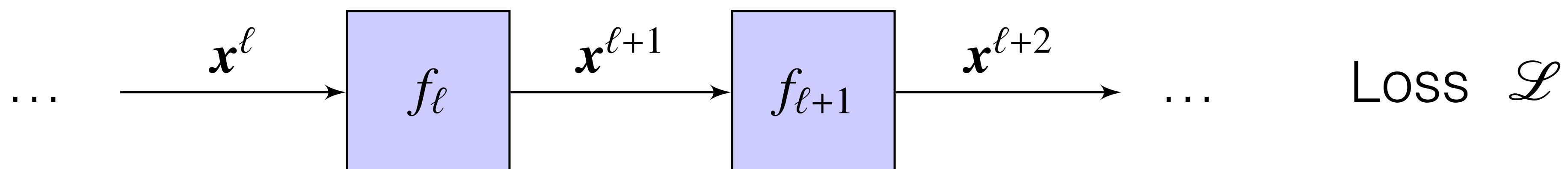


Differentiable tensor libraries with GPU/TPU/FPGA/... support



# The engine of deep learning: Back-Propagation algorithm

## computation graph



$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}^\ell} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{\ell+1}} \left( \frac{\partial \mathbf{x}^{\ell+1}}{\partial \mathbf{x}^\ell} \right)$$

Jacobian-Vector Product

$\overbrace{\hspace{10em}}$

**Computes gradients efficiently & accurately  
Via reverse mode automatic differentiation**

# Differentiable Programming

## Benefits

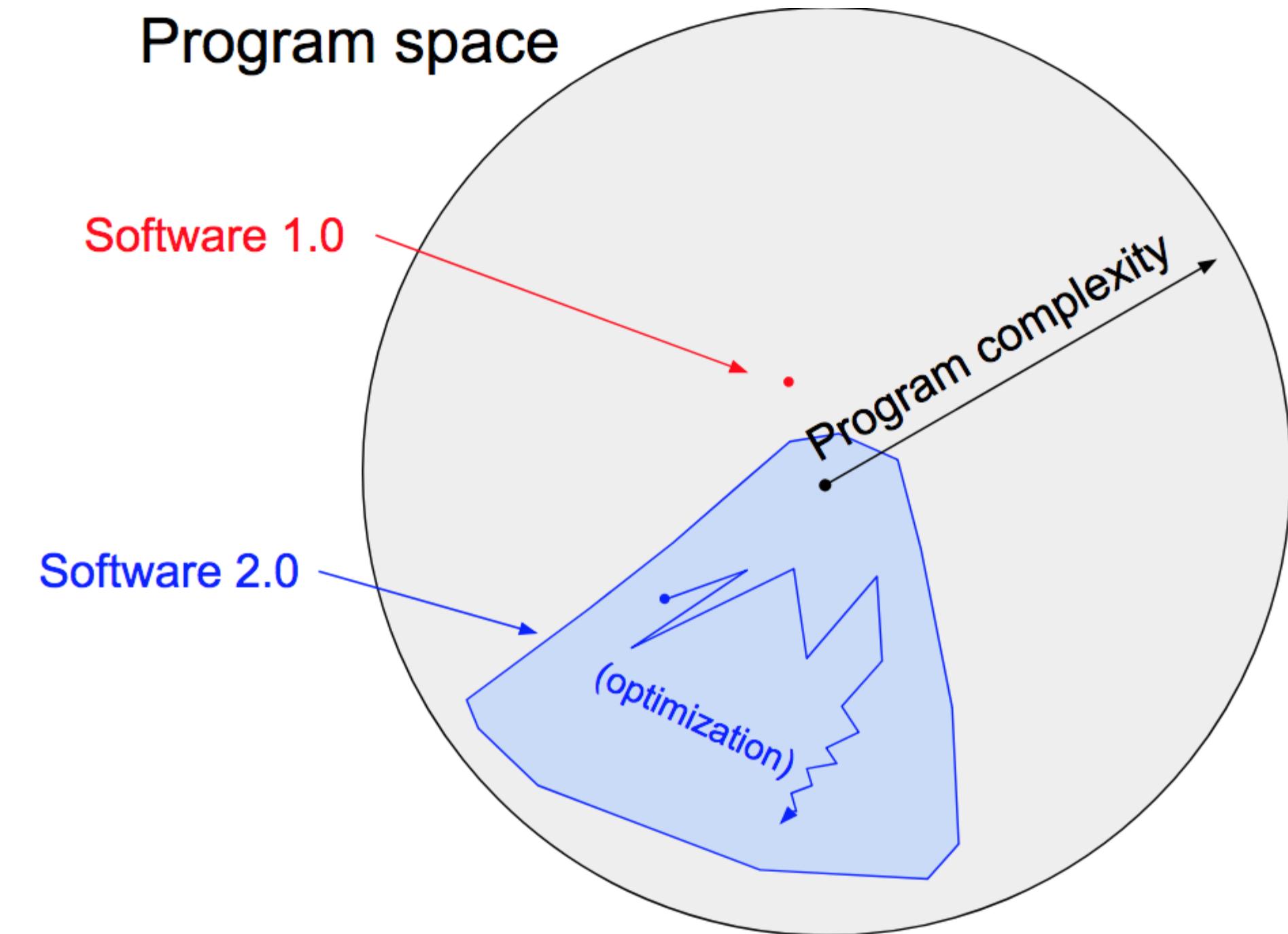
- Computationally homogeneous
- Simple to bake into silicon
- Constant running time
- Constant memory usage
- Highly portable & agile
- Modules can meld into an optimal whole
- Better than humans



**Andrej Karpathy**

Director of AI at Tesla. Previously Research Scientist at OpenAI and PhD student at Stanford. I like to train deep neural nets on large datasets.

<https://medium.com/@karpathy/software-2-0-a64152b37c35>



**Writing software 2.0 by gradient search in the program space**

# Differentiable Scientific Programming

- Most linear algebra operations (Eigen, SVD!) are differentiable
- Loop/Condition/Sort/Permutations are also differentiable
- ODE integrators are differentiable with  $O(1)$  memory
- Differentiable ray tracer and Differentiable fluid simulations
- Differentiable Monte Carlo/Tensor Network/Functional RG/  
Dynamical Mean Field Theory/Density Functional Theory...

**Differentiable programming is more than training neural networks**

# Differentiable Eigensolver

$$H\Psi = \Psi\Lambda$$

**Forward mode:** What happen if  $H = H + dH$  ? Perturbation theory

**Reverse mode:** How should I change  $H$  given  $\partial\mathcal{L}/\partial\Psi$  and  $\partial\mathcal{L}/\partial\Lambda$  ? Inverse perturbation theory

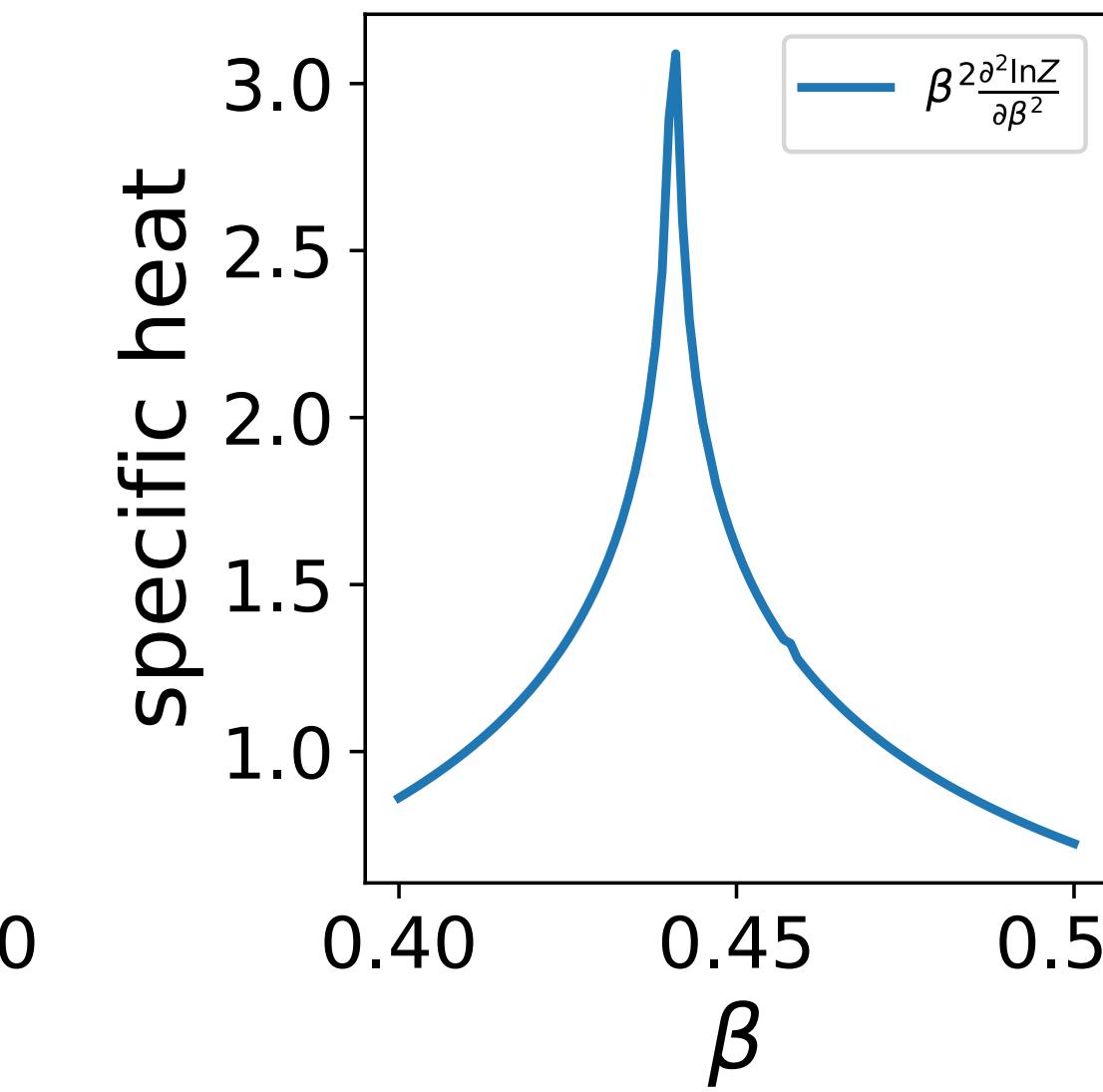
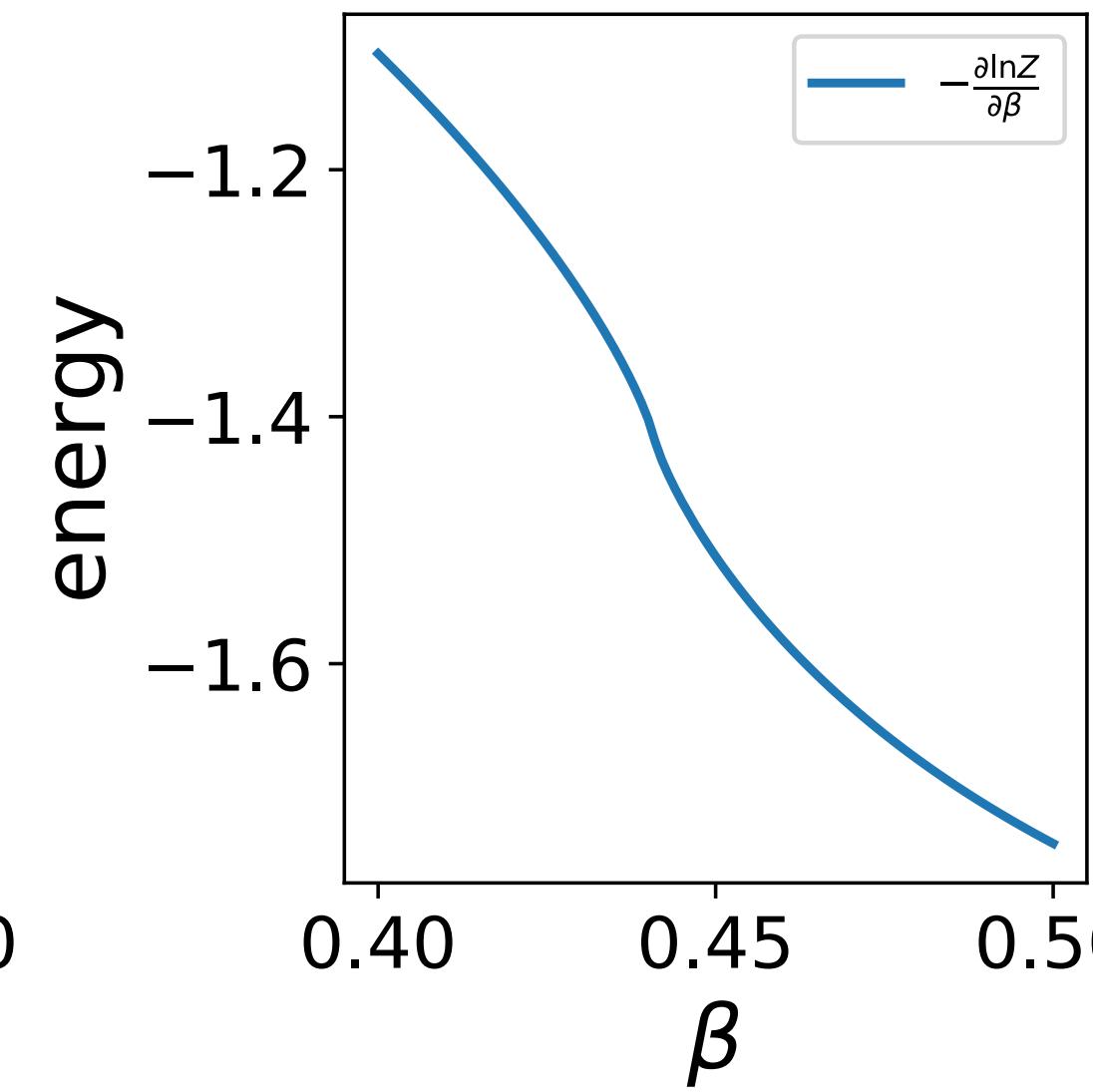
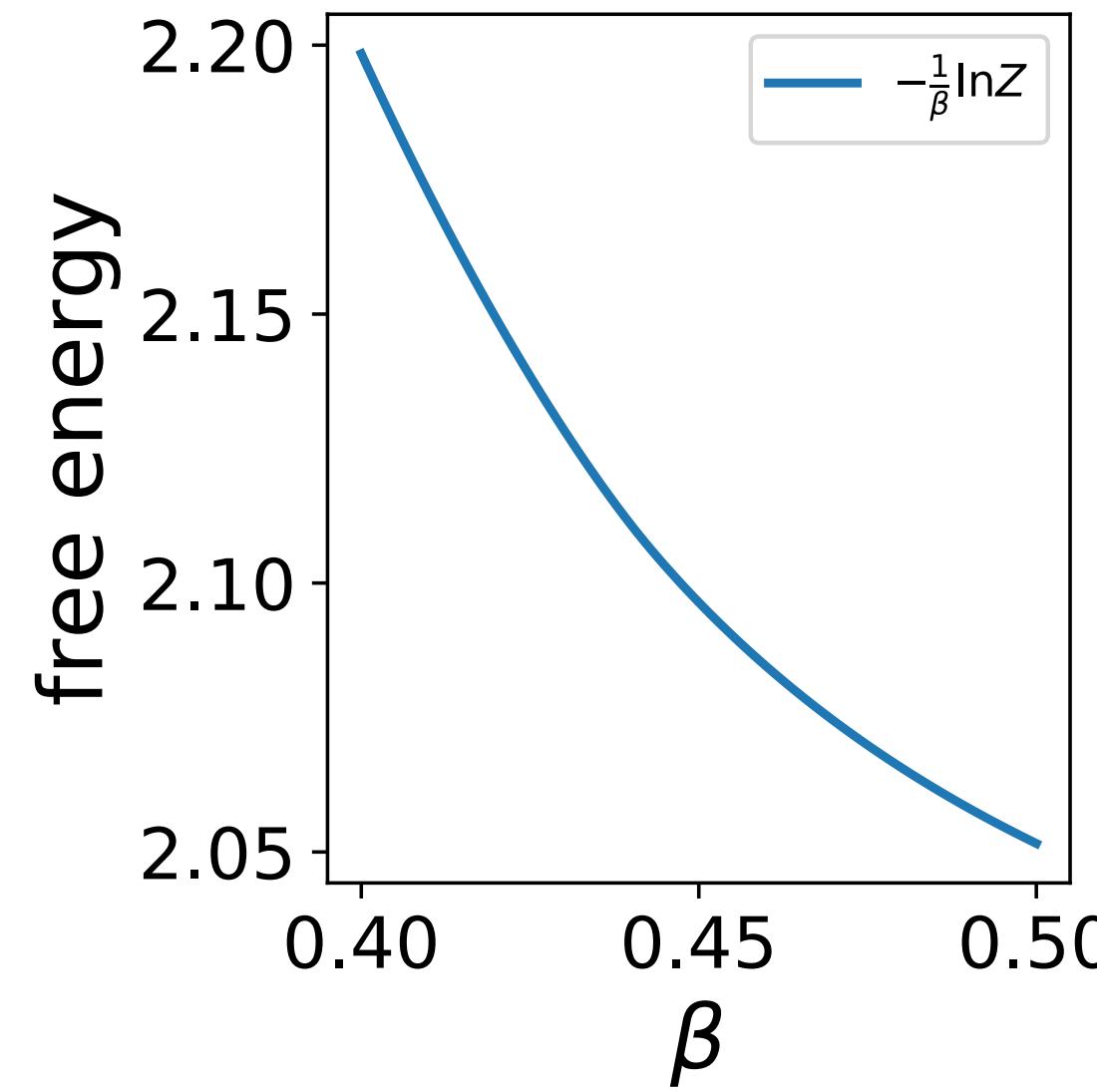
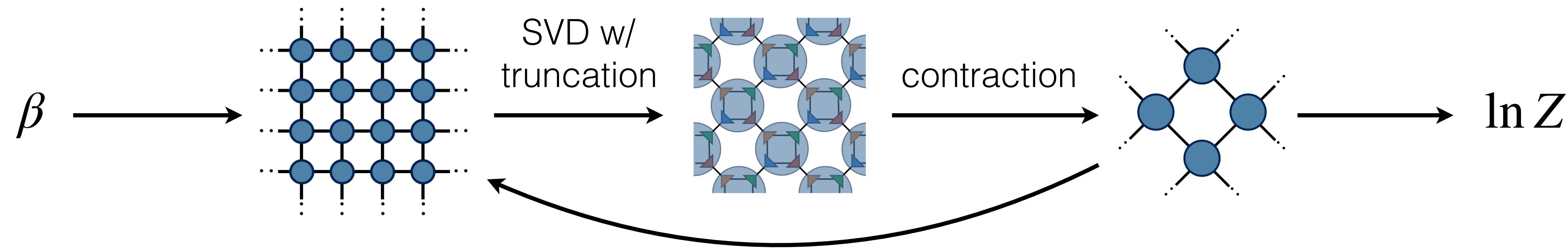
Hamiltonian engineering via differentiable programming



<https://github.com/wangleiphy/DL4CSRC/tree/master/2-ising>

# Differentiable Levin-Nave TRG

## computation graph

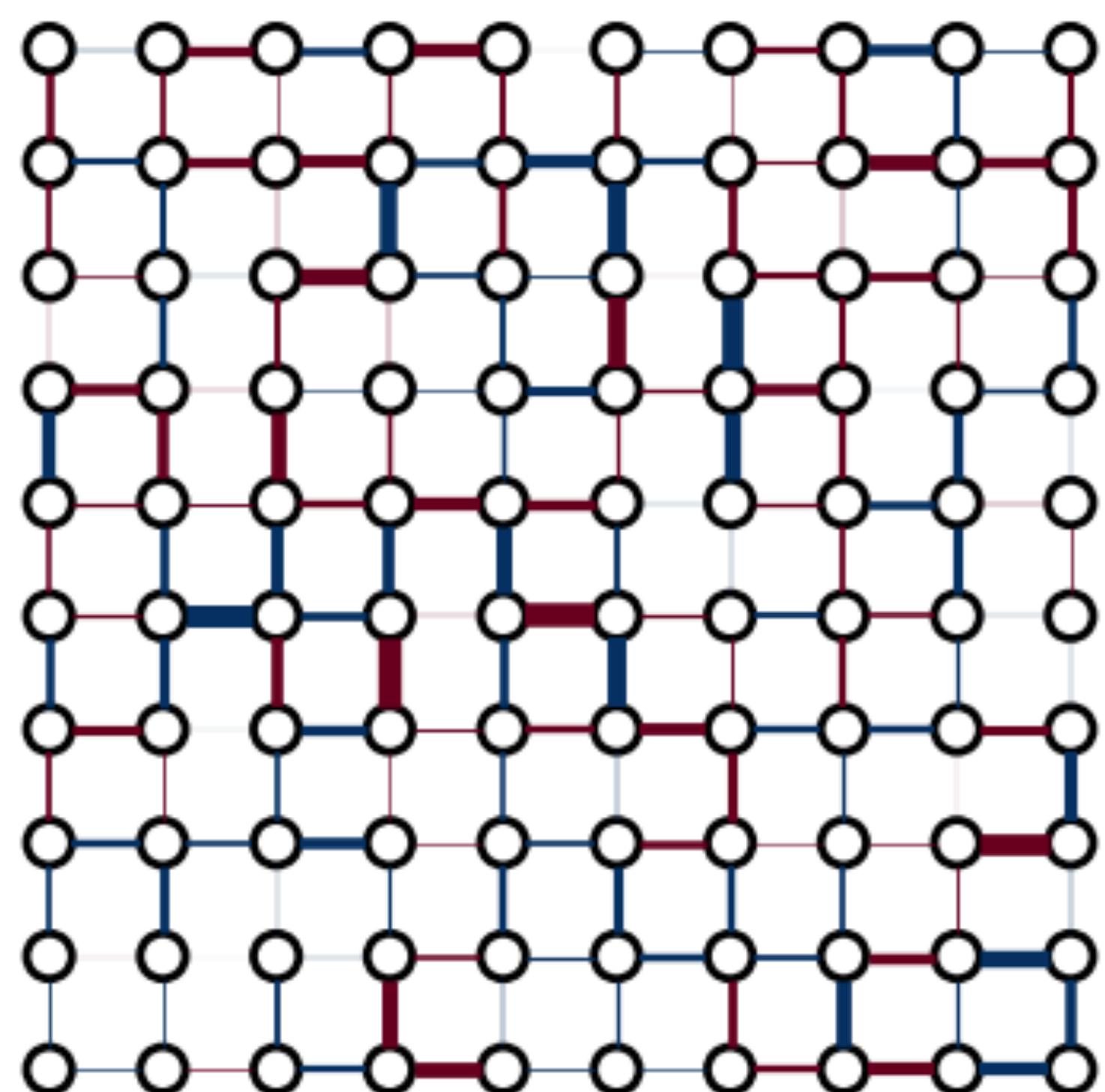


[https://github.com/  
wangleiphy/TRG](https://github.com/wangleiphy/TRG)

- Exact gradient
- Not finite-difference
- No additional codes
- Efforts comparable to the forward pass

**Automatic differentiation for high order gradients**

# Differentiable spin glass solver

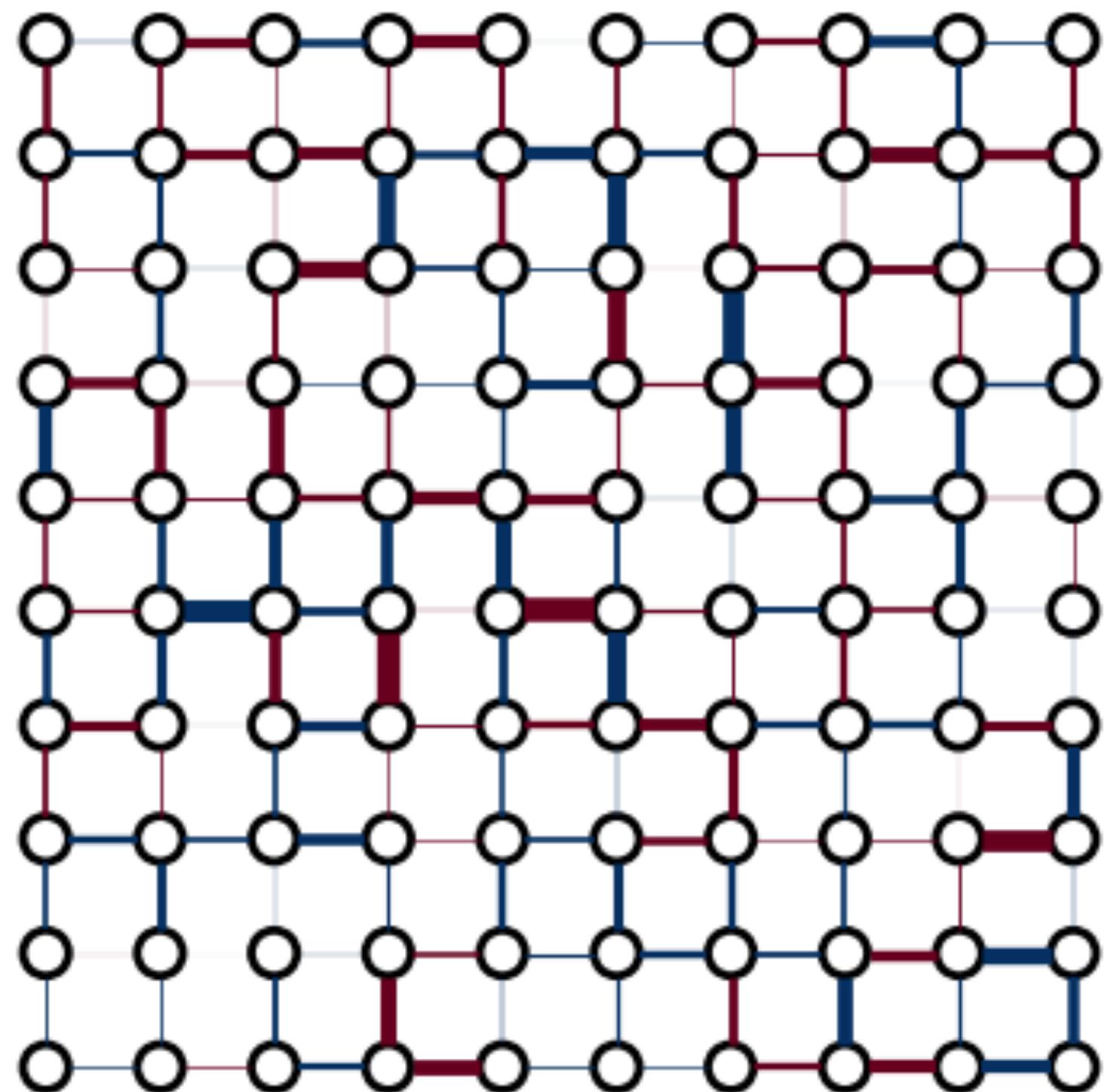


$K_{ij}$  random couplings

$$Z = \sum_{\{\sigma\}} \exp \left( \frac{1}{2} K_{ij} \sigma_i \sigma_j \right)$$

Contraction  
→  $\ln Z$

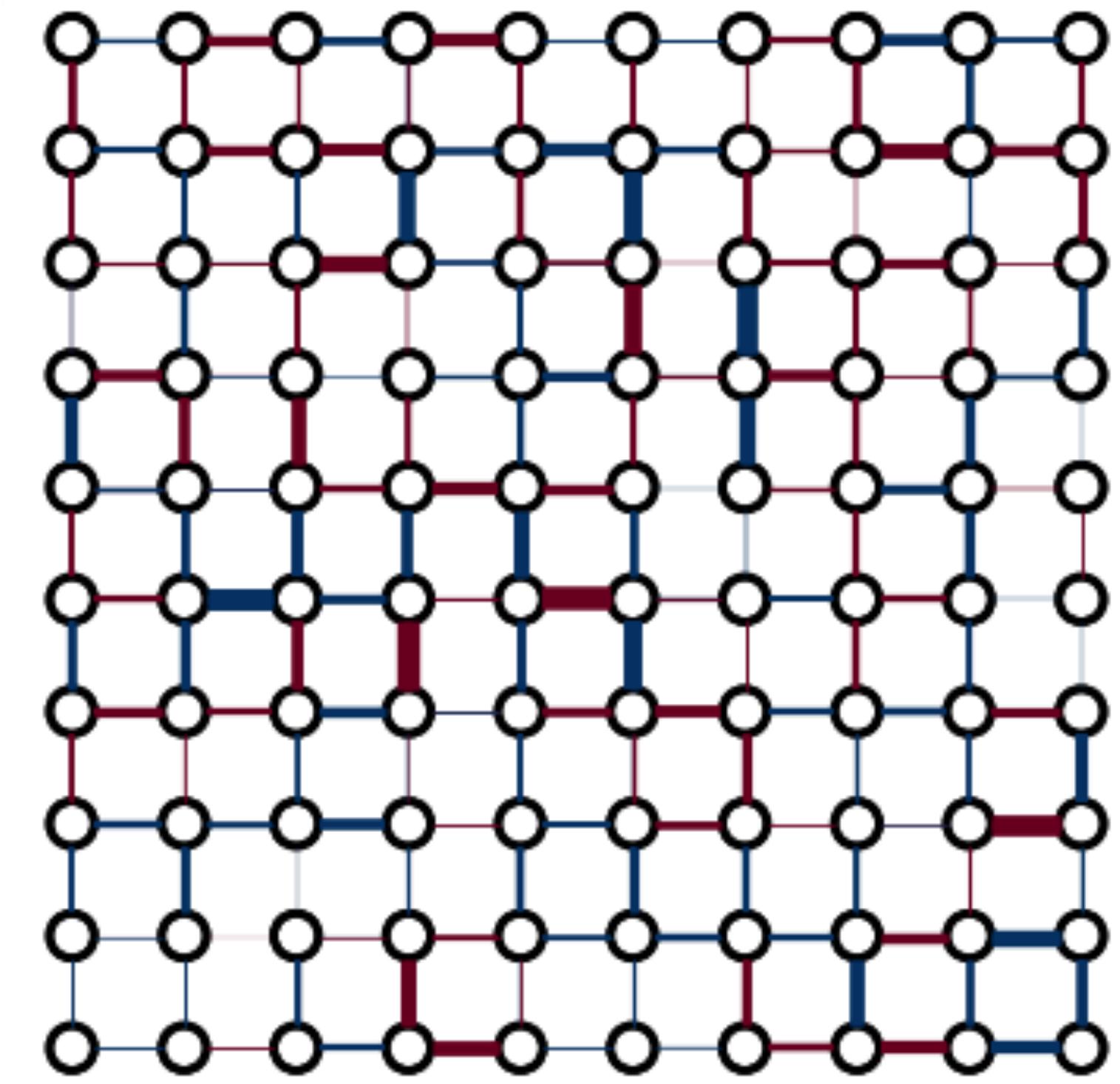
# Differentiable spin glass solver



$K_{ij}$  random couplings

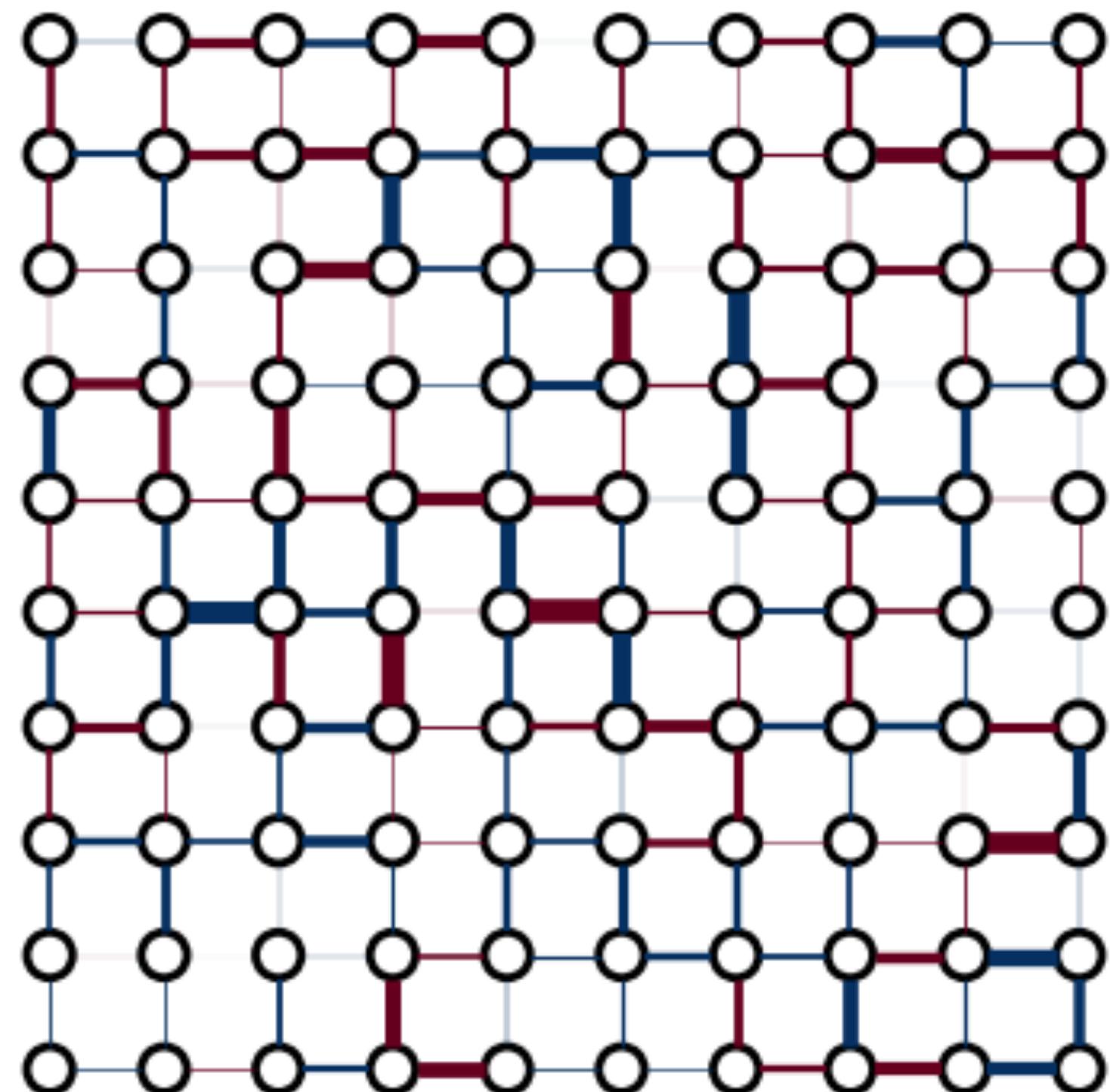
$$Z = \sum_{\{\sigma\}} \exp \left( \frac{1}{2} K_{ij} \sigma_i \sigma_j \right)$$

Contraction  $\longrightarrow \ln Z$  Differentiation  $\longrightarrow$

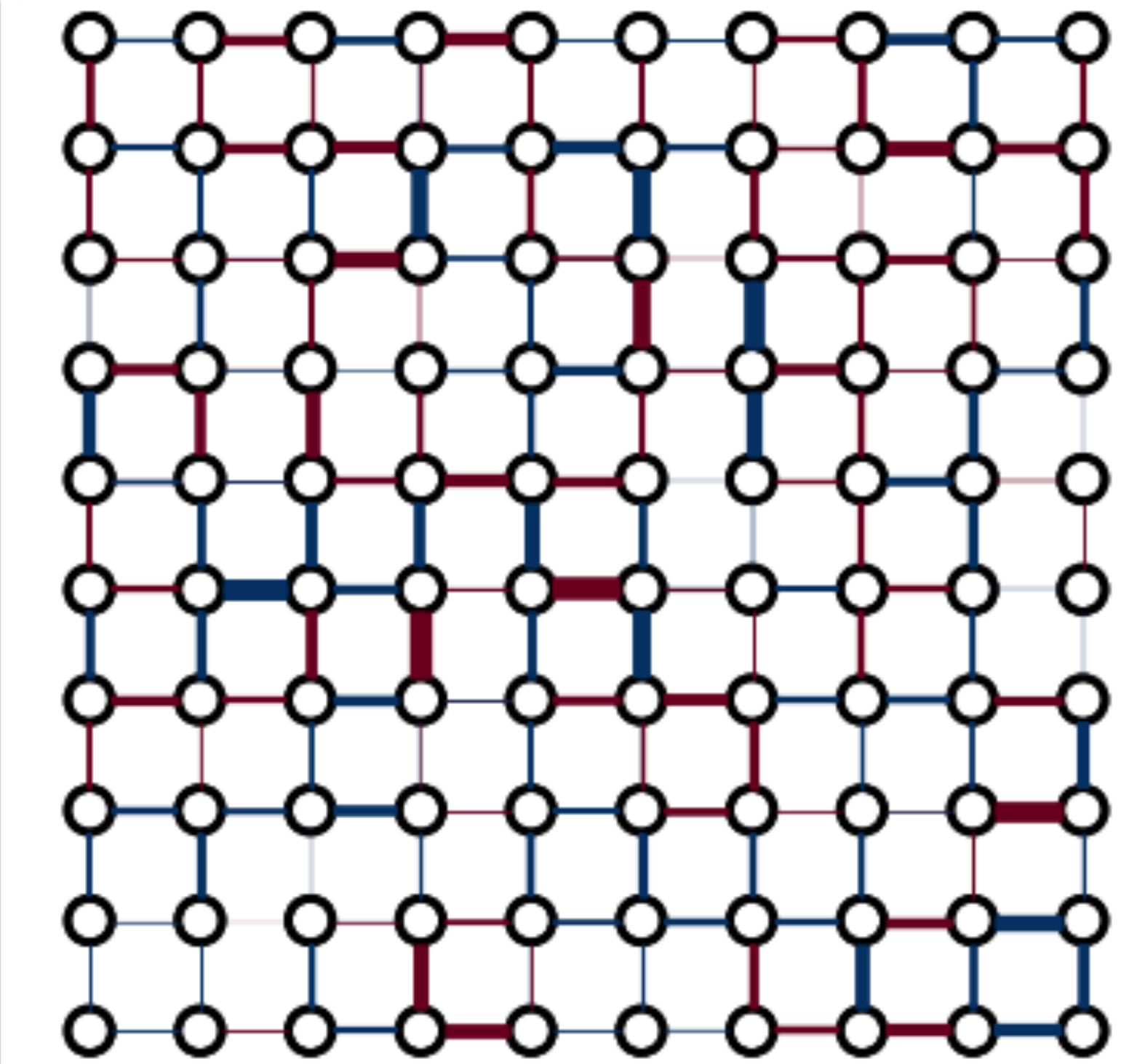
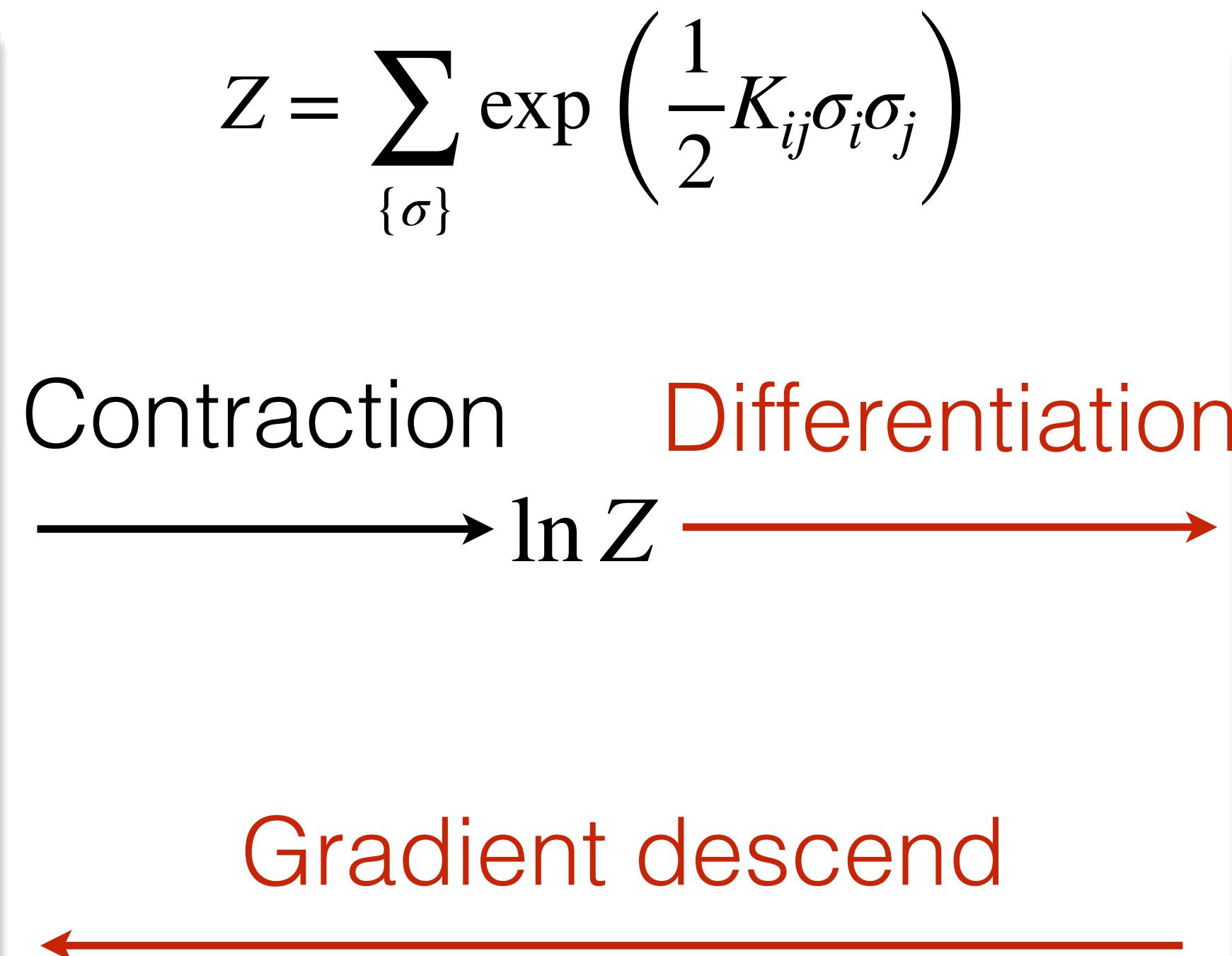


$$\frac{d \ln Z}{d K_{ij}} = \langle \sigma_i \sigma_j \rangle \text{ correlations}$$

# Differentiable spin glass solver



$K_{ij}$  random couplings



$$\frac{d \ln Z}{d K_{ij}} = \langle \sigma_i \sigma_j \rangle \text{ correlations}$$

**Tensor network solver for inverse Ising problems**

# Gradient based variational optimization

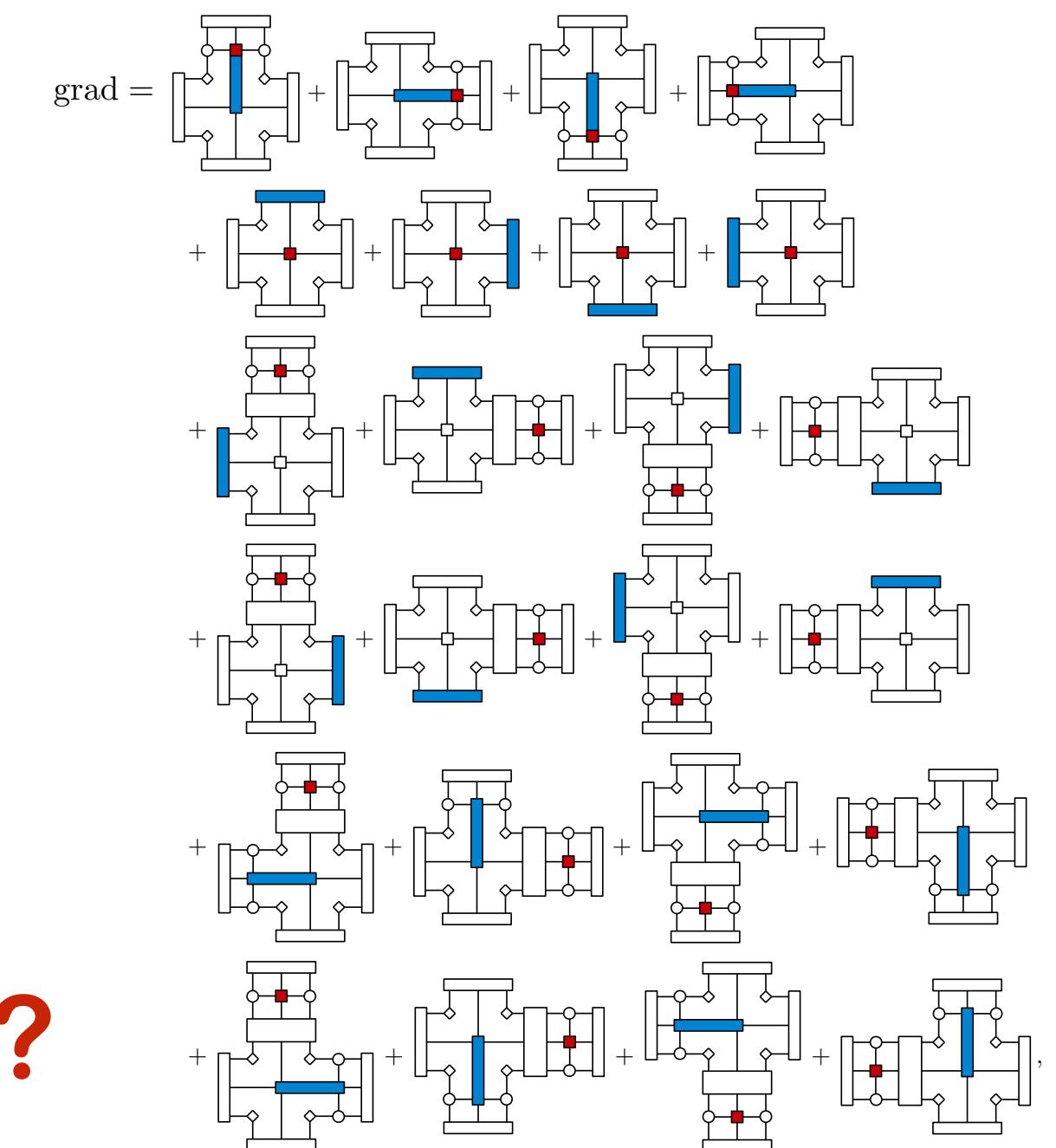
**Generative Modeling**

$$\mathcal{L} = \langle E(x) \rangle_{x \sim \mathcal{D}} + \ln Z$$

**Gradient optimization works fine**

**Variational ground state, why not ?**

$$\mathcal{L} = \ln \langle \Psi | \hat{H} | \Psi \rangle - \ln \langle \Psi | \Psi \rangle$$



Vanderstraeten et al,  
PRB 16'

**Human only cares about tensor contraction  
Differentiable programing takes care of the gradients**

# Thank You!

Song Cheng Jin-Guo Liu Jing Chen Zhiyuan Xie Pan Zhang Tao Xiang

